

Package ‘mctq’

May 9, 2022

Title Tools to Process the Munich ChronoType Questionnaire (MCTQ)

Version 0.2.0

Description A complete toolkit to process the Munich ChronoType Questionnaire (MCTQ) for its three versions (standard, micro, and shift). MCTQ is a quantitative and validated tool to assess chronotypes using peoples' sleep behavior, originally presented by Till Roenneberg, Anna Wirz-Justice, and Martha Merrow (2003, <[doi:10.1177/0748730402239679](https://doi.org/10.1177/0748730402239679)>).

biocViews Infrastructure, Preprocessing, Visualization

License MIT + file LICENSE

URL <https://docs.ropensci.org/mctq/>, <https://github.com/ropensci/mctq/>

BugReports <https://github.com/ropensci/mctq/issues/>

Depends R (>= 4.0)

Imports checkmate (>= 2.1.0), cli (>= 3.3.0), dplyr (>= 1.0.9), hms (>= 1.1.1), lifecycle (>= 1.0.1), lubridate (>= 1.8.0)

Suggests covr (>= 3.5.1), datasets (>= 4.0.0), ggplot2 (>= 3.3.6), grDevices (>= 4.0.0), knitr (>= 1.39), mockr (>= 0.2.0), readr (>= 2.1.2), rlang (>= 1.0.2), rmarkdown (>= 2.14), spelling (>= 2.2), stats (>= 4.0.0), testthat (>= 3.1.4), usethis (>= 2.1.5), utils (>= 4.0.0)

VignetteBuilder knitr

Config/testthat/edition 3

Encoding UTF-8

Language en-US

LazyData true

RoxygenNote 7.1.2

NeedsCompilation no

Author Daniel Vartanian [aut, cph, cre]
(<<https://orcid.org/0000-0001-7782-759X>>),
Ana Amelia Benedito-Silva [aut, sad]
(<<https://orcid.org/0000-0003-4976-2623>>),

Mario Pedrazzoli [aut, sad] (<<https://orcid.org/0000-0002-5257-591X>>),
 Jonathan Keane [rev] (<<https://orcid.org/0000-0001-7087-9776>>),
 Mario Andre Leocadio-Miguel [rev]
 (<<https://orcid.org/0000-0002-7248-3529>>),
 Interdisciplinary Sleep Research Group (GPSO) [fnd],
 University of Sao Paulo (USP) [fnd]

Maintainer Daniel Vartanian <danvartan@gmail.com>

Repository CRAN

Date/Publication 2022-05-09 08:00:02 UTC

R topics documented:

assign_date	3
cycle_time	5
fd	8
gu	10
le_week	12
micro_mctq	16
msf_sc	20
mst	24
napd	27
pretty_mctq	30
qplot_walk	31
random_mctq	34
raw_data	35
round_time	36
sd24	38
sdu	41
sd_overall	44
sd_week	47
shift_mctq	50
shorter_interval	59
sjl	62
sjl_sc	67
sjl_weighted	71
sloss_week	75
so	78
std_mctq	81
sum_time	88
tbt	92

Index **96**

assign_date	<i>Assign dates to two sequential hours</i>
-------------	---

Description

[Maturing]

assign_date() assign dates to two sequential hours. It can facilitate time arithmetic by locating time values without a date reference on a timeline.

Usage

```
assign_date(start, end, ambiguity = 0)
```

Arguments

start, end	An hms or POSIXt object indicating the start or end hour.
ambiguity	(optional) a numeric or NA value to instruct assign_date() on how to deal with ambiguities. See the Details section to learn more (default: 0).

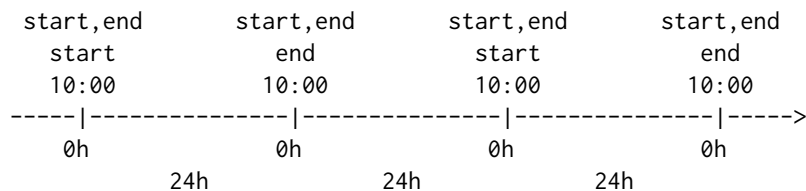
Details

Class requirements:

The mctq package works with a set of object classes specially created to hold time values. These classes can be found in the [lubridate](#) and [hms](#) packages. Please refer to those package documents to learn more about them.

ambiguity argument:

In cases when start is equal to end, there are two possibilities of intervals between the two hours (ambiguity). That's because start and end can be at the same point in time or they can distance themselves by one day, considering a two-day timeline.



You must instruct assign_date() on how to deal with this problem if it occurs. There are three options to choose.

- ambiguity = 0: to consider the interval between start and end as 0 hours, i.e., start and end are located at the same point in time (default).
- ambiguity = 24: to consider the interval between start and end as 24 hours, i.e., start and end distance themselves by one day.
- ambiguity = NA: to disregard these cases, assigning NA as value.

Base date and timezone:

assign_date() uses the **Unix epoch** (1970-01-01) date as the start date for creating intervals. The output will always have "UTC" set as timezone. Learn more about time zones in [?timezone](#).

POSIXt objects:

POSIXt objects passed as argument to start or end will be stripped of their dates. Only the time will be considered.

Both POSIXct and POSIXlt are objects that inherits the class POSIXt. Learn more about it in [?DateTimeClasses](#).

NA values:

assign_date() will return an **Interval** NA-NA if start or end are NA.

Value

A start–end **Interval** object.

See Also

Other utility functions: [cycle_time\(\)](#), [pretty_mctq\(\)](#), [qplot_walk\(\)](#), [random_mctq\(\)](#), [raw_data\(\)](#), [round_time\(\)](#), [shorter_interval\(\)](#), [sum_time\(\)](#)

Examples

```
## Scalar example

start <- hms::parse_hms("23:11:00")
end <- hms::parse_hms("05:30:00")
assign_date(start, end)
#> [1] 1970-01-01 23:11:00 UTC--1970-01-02 05:30:00 UTC # Expected

start <- hms::parse_hms("10:15:00")
end <- hms::parse_hms("13:25:00")
assign_date(start, end)
#> [1] 1970-01-01 10:15:00 UTC--1970-01-01 13:25:00 UTC # Expected

start <- hms::parse_hms("05:42:00")
end <- hms::as_hms(NA)
assign_date(start, end)
#> [1] NA--NA # Expected

## Vector example

start <- c(hms::parse_hm("09:45"), hms::parse_hm("20:30"))
end <- c(hms::parse_hm("21:15"), hms::parse_hm("04:30"))
assign_date(start, end)
#> [1] 1970-01-01 09:45:00 UTC--1970-01-01 21:15:00 UTC # Expected
#> [2] 1970-01-01 20:30:00 UTC--1970-01-02 04:30:00 UTC # Expected

## To assign a 24 hours interval to ambiguities
```

```

start <- lubridate::as_datetime("1985-01-15 12:00:00")
end <- lubridate::as_datetime("2020-09-10 12:00:00")
assign_date(start, end, ambiguity = 24)
#> [1] 1970-01-01 12:00:00 UTC--1970-01-02 12:00:00 UTC # Expected

```

cycle_time

Cycle time objects

Description

[Maturing]

cycle_time() cycles time span objects in a predetermined cycle length, adapting linear time objects into a circular time frame.

Usage

```
cycle_time(time, cycle, reverse = TRUE)
```

Arguments

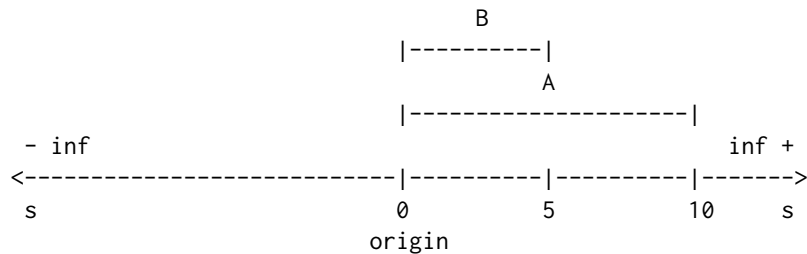
time	An object belonging to one of the following classes: numeric , Duration , difftime , or hms .
cycle	A numeric or Duration object of length 1, equal or greater than 0, indicating the cycle length in seconds. See the Details section to learn more.
reverse	(optional) a logical value indicating if the function must use a reverse cycle for negative values in time. See the Details section to learn more (default: TRUE).

Details

Linear versus circular time:

Time can have different "shapes".

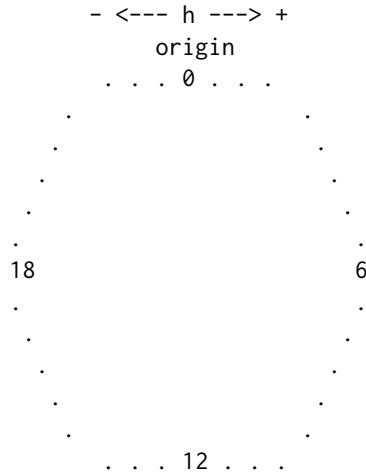
If the objective is to measure the duration (time span) of an event, time is usually measured considering a linear frame, with a fixed point of **origin**. In this context, the time value distance itself to infinity in relation to the origin.



$$A + B = 10 + 5 = 15s$$

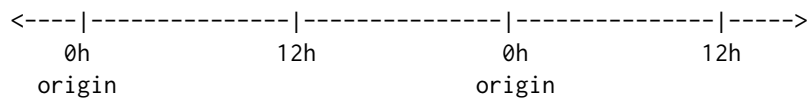
But that's not the only possible "shape" of time, as it can also be measured in other contexts.

In a "time of day" context, the time will be linked to the rotation of the earth, "resetting" when a new rotation cycle starts. That brings a different kind of shape to time: a circular shape. With this shape the time value encounters the origin at the beginning and end of each cycle.



$$18 + 6 = 0h$$

If we transpose this circular time frame to a linear one, it would look like this:



Note that now the origin is not fix, but cyclical.

`cycle_time()` operates by converting linear time objects using a circular approach relative to the cycle length (e.g, `cycle = 86400` (1 day)).

Fractional time:

`cycle_time()` uses the `%` operator to cycle values. Hence, it can be subject to catastrophic loss of accuracy if time is fractional and much larger than `cycle`. A warning is given if this is detected.

`%` is a builtin R function that operates like this:

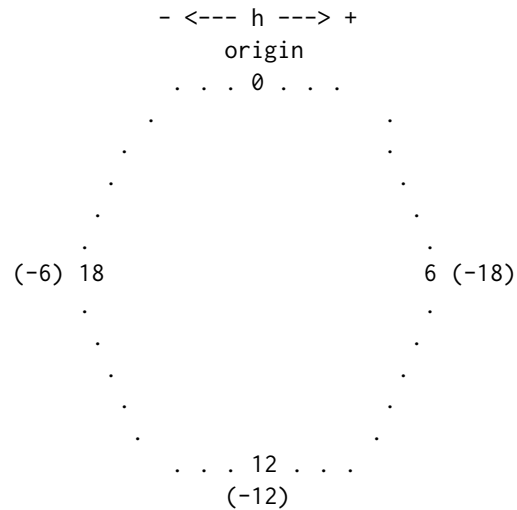
```
function(a, b) {
  a - floor(a / b) * b
}
```

Negative time cycling:

If time have a negative value and `reverse == FALSE`, `cycle_time()` will perform the cycle considering the absolute value of time and return the result with a negative signal.

However, If time have a negative value and `reverse == TRUE` (default), `cycle_time()` will perform the cycle in reverse, relative to its origin.

Example: If you have a -30h time span in a reversed cycle of 24h, the result will be 18h. By removing the full cycles of -30h you will get -6h (-30 + 24), and -6h relative to the origin will be 18h.

**Period objects:**

Period objects are a special type of object developed by the [lubridate](#) team that represents "human units", ignoring possible timeline irregularities. That is to say that 1 day as **Period** can have different time spans, when looking to a timeline after a irregularity event.

Since the time span of a **Period** object can fluctuate, `cycle_time()` don't accept this kind of object. You can transform it to a **Duration** object and still use the function, but beware that this can produce errors.

Learn more about **Period** objects in the [Dates and times](#) chapter of Wickham & Grolemund book (n.d.).

Value

The same type of object of time cycled with the `cycle` parameter.

References

Wickham, H., & Grolemund, G. (n.d.). *R for data science*. Sebastopol, CA: O'Reilly Media.
<https://r4ds.had.co.nz>

See Also

Other utility functions: [assign_date\(\)](#), [pretty_mctq\(\)](#), [qplot_walk\(\)](#), [random_mctq\(\)](#), [raw_data\(\)](#), [round_time\(\)](#), [shorter_interval\(\)](#), [sum_time\(\)](#)

Examples

```
## Scalar example

time <- lubridate::dhours(25)
cycle <- lubridate::ddays(1)
cycle_time(time, cycle)
#> [1] "3600s (~1 hours)" # Expected
```

```

time <- lubridate::dhours(-25)
cycle <- lubridate::ddays(1)
reverse <- FALSE
cycle_time(time, cycle, reverse)
#> [1] "-3600s (~-1 hours)" # Expected

time <- lubridate::dhours(-25)
cycle <- lubridate::ddays(1)
reverse <- TRUE
cycle_time(time, cycle, reverse)
#> [1] "82800s (~23 hours)" # Expected

## Vector example

time <- c(lubridate::dmonths(24), lubridate::dmonths(13))
cycle <- lubridate::dyears(1)
cycle_time(time, cycle)
#> [1] "0s" "2629800s (~4.35 weeks)" # Expected

time <- c(lubridate::dmonths(24), lubridate::dmonths(-13))
cycle <- lubridate::dyears(1)
reverse <- FALSE
cycle_time(time, cycle, reverse)
#> [1] "0s" "-2629800s (~-4.35 weeks)" # Expected

time <- c(lubridate::dmonths(24), lubridate::dmonths(-13))
cycle <- lubridate::dyears(1)
reverse <- TRUE
cycle_time(time, cycle, reverse)
#> [1] "0s" "28927800s (~47.83 weeks)" # Expected

```

 fd

Compute MCTQ work-free days

Description

[Maturing]

fd() computes the **number of work-free days per week** for standard and micro versions of the Munich ChronoType Questionnaire (MCTQ).

Usage

```
fd(wd)
```

Arguments

wd An [integerish numeric](#) object or an [integer](#) object corresponding to the **number of workdays per week** from a standard or micro version of the MCTQ questionnaire.

Details

Standard MCTQ functions were created following the guidelines in Roenneberg, Wirz-Justice, & Merrow (2003), Roenneberg, Allebrandt, Merrow, & Vetter (2012), and from The Worldwide Experimental Platform (theWeP, n.d.).

μ **MCTQ** functions were created following the guidelines in Ghotbi et al. (2020), in addition to the guidelines used for the standard MCTQ.

MCTQ^{Shift} functions were created following the guidelines in Juda, Vetter, & Roenneberg (2013), in addition to the guidelines used for the standard MCTQ.

See the References section to learn more.

Value

An `integer` object corresponding to the difference between the number of days in a week (7) and the number of workdays (wd).

Guidelines

Roenneberg, Allebrandt, Merrow, & Vetter (2012) and The Worldwide Experimental Platform (n.d.) guidelines for `fd()` (*FD*) computation are as follows.

$$7 - WD$$

Where:

- *WD* = Number of workdays ("I have a regular work schedule and work ___ days per week").

References

- Ghotbi, N., Pilz, L. K., Winnebeck, E. C., Vetter, C., Zerbini, G., Lenssen, D., Frighetto, G., Salamanca, M., Costa, R., Montagnese, S., & Roenneberg, T. (2020). The μ MCTQ: an ultra-short version of the Munich ChronoType Questionnaire. *Journal of Biological Rhythms*, 35(1), 98-110. doi: [10.1177/0748730419886986](https://doi.org/10.1177/0748730419886986).
- Juda, M., Vetter, C., & Roenneberg, T. (2013). The Munich ChronoType Questionnaire for shift-workers (MCTQ^{Shift}). *Journal of Biological Rhythms*, 28(2), 130-140. doi: [10.1177/0748730412475041](https://doi.org/10.1177/0748730412475041).
- Roenneberg T., Allebrandt K. V., Merrow M., & Vetter C. (2012). Social jetlag and obesity. *Current Biology*, 22(10), 939-43. doi: [10.1016/j.cub.2012.03.038](https://doi.org/10.1016/j.cub.2012.03.038).
- Roenneberg, T., Wirz-Justice, A., & Merrow, M. (2003). Life between clocks: daily temporal patterns of human chronotypes. *Journal of Biological Rhythms*, 18(1), 80-90. doi: [10.1177/0748730402239679](https://doi.org/10.1177/0748730402239679).
- The Worldwide Experimental Platform (n.d.). MCTQ. <https://www.thewep.org/documentations/mctq/>

See Also

Other MCTQ functions: `gu()`, `le_week()`, `msf_sc()`, `msl()`, `napd()`, `sd24()`, `sd_overall()`, `sd_week()`, `sdu()`, `sjl_sc()`, `sjl_weighted()`, `sjl()`, `so()`, `tbt()`

Examples

```
## Scalar example

fd(5)
#> [1] 2 # Expected
fd(4)
#> [1] 3 # Expected
fd(as.numeric(NA))
#> [1] NA # Expected

## Vector example

fd(0:7)
#> [1] 7 6 5 4 3 2 1 0 # Expected
fd(c(1, NA))
#> [1] 6 NA # Expected
```

```
gu          Compute MCTQ local time of getting out of bed
```

Description

[Maturing]

gu() computes the **local time of getting out of bed** for standard and shift versions of the Munich ChronoType Questionnaire (MCTQ).

Usage

```
gu(se, si)
```

Arguments

se	An hms object corresponding to the local time of sleep end from a standard or shift version of the MCTQ questionnaire.
si	A Duration object corresponding to the " sleep inertia " or time to get up from a standard or shift version of the MCTQ questionnaire.

Details

Standard MCTQ functions were created following the guidelines in Roenneberg, Wirz-Justice, & Mewes (2003), Roenneberg, Allebrandt, Mewes, & Vetter (2012), and from The Worldwide Experimental Platform (theWeP, n.d.).

μ **MCTQ** functions were created following the guidelines in Ghotbi et al. (2020), in addition to the guidelines used for the standard MCTQ.

MCTQ^{Shift} functions were created following the guidelines in Juda, Vetter, & Roenneberg (2013), in addition to the guidelines used for the standard MCTQ.

See the References section to learn more.

Class requirements:

The `mctq` package works with a set of object classes specially created to hold time values. These classes can be found in the `lubridate` and `hms` packages. Please refer to those package documentations to learn more about them.

Rounding and fractional time:

Some operations may produce an output with fractional time (e.g., "19538.3828571429s (~5.43 hours)", 01:15:44.505). If you want, you can round it with `round_time()`.

Our recommendation is to avoid rounding, but, if you do, make sure that you only round your values after all computations are done. That way you avoid **round-off errors**.

Value

An `hms` object corresponding to the vectorized sum of `se` and `si` in a circular time frame of 24 hours.

Guidelines

Roenneberg, Allebrandt, Merrow, & Vetter (2012), Juda, Vetter, & Roenneberg (2013), and The Worldwide Experimental Platform (n.d.) guidelines for `gu()` (*GU*) computation are as follows.

Notes:

- This computation must be applied to each section of the questionnaire.
- `MCTQShift` uses *TGU* (time to get up) instead of *SI* (sleep inertia). For the purpose of this computation, both represent the same thing.
- If you are visualizing this documentation in plain text, you may have some trouble understanding the equations. You can see this documentation on the package [website](#).

For standard and micro versions of the MCTQ:

$$SE_{W/F} + SI_{W/F}$$

Where:

- $SE_{W/F}$ = Local time of sleep end on work **or** work-free days.
- $SI_{W/F}$ = Sleep inertia on work **or** work-free days ("after ___ min, I get up").

* W = Workdays; F = Work-free days.

For the shift version of the MCTQ:

$$SE_{W/F}^{M/E/N} + TGU_{W/F}^{M/E/N}$$

Where:

- $SE_{W/F}^{M/E/N}$ = Local time of sleep end between two days in a particular shift **or** between two free days after a particular shift.
- $TGU_{W/F}^{M/E/N}$ = Time to get up after sleep end between two days in a particular shift **or** between two free days after a particular shift ("after ___ min, I get up").

* W = Workdays; F = Work-free days, M = Morning shift; E = Evening shift; N = Night shift.

References

- Ghotbi, N., Pilz, L. K., Winnebeck, E. C., Vetter, C., Zerbini, G., Lenssen, D., Frighetto, G., Salamanca, M., Costa, R., Montagnese, S., & Roenneberg, T. (2020). The μ MCTQ: an ultra-short version of the Munich ChronoType Questionnaire. *Journal of Biological Rhythms*, 35(1), 98-110. doi: [10.1177/0748730419886986](https://doi.org/10.1177/0748730419886986).
- Juda, M., Vetter, C., & Roenneberg, T. (2013). The Munich ChronoType Questionnaire for shift-workers (MCTQ^{Shift}). *Journal of Biological Rhythms*, 28(2), 130-140. doi: [10.1177/0748730412475041](https://doi.org/10.1177/0748730412475041).
- Roenneberg T., Allebrandt K. V., Merrow M., & Vetter C. (2012). Social jetlag and obesity. *Current Biology*, 22(10), 939-43. doi: [10.1016/j.cub.2012.03.038](https://doi.org/10.1016/j.cub.2012.03.038).
- Roenneberg, T., Wirz-Justice, A., & Merrow, M. (2003). Life between clocks: daily temporal patterns of human chronotypes. *Journal of Biological Rhythms*, 18(1), 80-90. doi: [10.1177/0748730402239679](https://doi.org/10.1177/0748730402239679).
- The Worldwide Experimental Platform (n.d.). MCTQ. <https://www.thewep.org/documentations/mctq/>

See Also

Other MCTQ functions: [fd\(\)](#), [le_week\(\)](#), [msf_sc\(\)](#), [msl\(\)](#), [napd\(\)](#), [sd24\(\)](#), [sd_overall\(\)](#), [sd_week\(\)](#), [sdu\(\)](#), [sjl_sc\(\)](#), [sjl_weighted\(\)](#), [sjl\(\)](#), [so\(\)](#), [tbt\(\)](#)

Examples

```
## Scalar example

gu(hms::parse_hm("08:00"), lubridate::dminutes(10))
#> 08:10:00 # Expected
gu(hms::parse_hm("11:45"), lubridate::dminutes(90))
#> 13:15:00 # Expected
gu(hms::as_hms(NA), lubridate::dminutes(90))
#> NA # Expected

## Vector example

se <- c(hms::parse_hm("12:30"), hms::parse_hm("23:45"))
si <- c(lubridate::dminutes(10), lubridate::dminutes(70))
gu(se, si)
#> 12:40:00 # Expected
#> 00:55:00 # Expected
```

le_week

Compute MCTQ average weekly light exposure

Description

[Maturing]

`le_week()` computes the **average weekly light exposure** for the standard version of the Munich ChronoType Questionnaire (MCTQ).

Usage

```
le_week(le_w, le_f, wd)
```

Arguments

le_w	A Duration object corresponding to the light exposure on workdays from a standard version of the MCTQ questionnaire.
le_f	A Duration object corresponding to the light exposure on work-free days from a standard version of the MCTQ questionnaire.
wd	An integerish numeric object or an integer object corresponding to the number of workdays per week from a standard version of the MCTQ questionnaire.

Details

Standard MCTQ functions were created following the guidelines in Roenneberg, Wirz-Justice, & Meroow (2003), Roenneberg, Allebrandt, Meroow, & Vetter (2012), and from The Worldwide Experimental Platform (theWeP, n.d.).

μ MCTQ functions were created following the guidelines in Ghotbi et al. (2020), in addition to the guidelines used for the standard MCTQ.

MCTQ^{Shift} functions were created following the guidelines in Juda, Vetter, & Roenneberg (2013), in addition to the guidelines used for the standard MCTQ.

See the References section to learn more.

Class requirements:

The mctq package works with a set of object classes specially created to hold time values. These classes can be found in the [lubridate](#) and [hms](#) packages. Please refer to those package documents to learn more about them.

Rounding and fractional time:

Some operations may produce an output with fractional time (e.g., "19538.3828571429s (~5.43 hours)", 01:15:44.505). If you want, you can round it with [round_time\(\)](#).

Our recommendation is to avoid rounding, but, if you do, make sure that you only round your values after all computations are done. That way you avoid **round-off errors**.

Value

A [Duration](#) object corresponding to the vectorized weighted mean of le_w and le_f with wd and fd(wd) as weights.

Guidelines

Roenneberg, Allebrandt, Meroow, & Vetter (2012) and The Worldwide Experimental Platform (n.d.) guidelines for le_week() (LE_{week}) computation are as follows.

Notes:

- The average weekly light exposure is the weighted average of the light exposure on work and work-free days in a week.

- If you are visualizing this documentation in plain text, you may have some trouble understanding the equations. You can see this documentation on the package [website](#).

Computation:

$$\frac{(LE_W \times WD) + (LE_F \times FD)}{7}$$

Where:

- LE_W = Light exposure on workdays.
- LE_F = Light exposure on work-free days.
- WD = Number of workdays per week ("I have a regular work schedule and work ___ days per week").
- FD = Number of work-free days per week.

* W = Workdays; F = Work-free days.

References

- Ghotbi, N., Pilz, L. K., Winnebeck, E. C., Vetter, C., Zerbini, G., Lenssen, D., Frighetto, G., Salamanca, M., Costa, R., Montagnese, S., & Roenneberg, T. (2020). The μ MCTQ: an ultra-short version of the Munich ChronoType Questionnaire. *Journal of Biological Rhythms*, 35(1), 98-110. doi: [10.1177/0748730419886986](https://doi.org/10.1177/0748730419886986).
- Juda, M., Vetter, C., & Roenneberg, T. (2013). The Munich ChronoType Questionnaire for shiftworkers (MCTQ^{Shift}). *Journal of Biological Rhythms*, 28(2), 130-140. doi: [10.1177/0748730412475041](https://doi.org/10.1177/0748730412475041).
- Roenneberg T., Allebrandt K. V., Mewes M., & Vetter C. (2012). Social jetlag and obesity. *Current Biology*, 22(10), 939-43. doi: [10.1016/j.cub.2012.03.038](https://doi.org/10.1016/j.cub.2012.03.038).
- Roenneberg, T., Wirz-Justice, A., & Mewes, M. (2003). Life between clocks: daily temporal patterns of human chronotypes. *Journal of Biological Rhythms*, 18(1), 80-90. doi: [10.1177/0748730402239679](https://doi.org/10.1177/0748730402239679).
- The Worldwide Experimental Platform (n.d.). MCTQ. <https://www.thewep.org/documentations/mctq/>

See Also

Other MCTQ functions: [fd\(\)](#), [gu\(\)](#), [msf_sc\(\)](#), [msl\(\)](#), [napd\(\)](#), [sd24\(\)](#), [sd_overall\(\)](#), [sd_week\(\)](#), [sdu\(\)](#), [sjl_sc\(\)](#), [sjl_weighted\(\)](#), [sjl\(\)](#), [so\(\)](#), [tbt\(\)](#)

Examples

```
## Scalar example

le_w <- lubridate::dhours(1.5)
le_f <- lubridate::dhours(3.7)
wd <- 5
le_week(le_w, le_f, wd)
#> [1] "7662.85714285714s (~2.13 hours)" # Expected
```

```

le_w <- lubridate::dhours(3)
le_f <- lubridate::dhours(1.5)
wd <- 6
le_week(le_w, le_f, wd)
#> [1] "10028.5714285714s (~2.79 hours)" # Expected

le_w <- lubridate::dhours(5.6)
le_f <- lubridate::as.duration(NA)
wd <- 3
le_week(le_w, le_f, wd)
#> [1] NA # Expected

## Vector example

le_w <- c(lubridate::dhours(3), lubridate::dhours(2.45))
le_f <- c(lubridate::dhours(3), lubridate::dhours(3.75))
wd <- c(4, 5)
le_week(le_w, le_f, wd)
#> [1] "10800s (~3 hours)" # Expected
#> [2] "10157.1428571429s (~2.82 hours)" # Expected

## Checking second output from vector example

if (requireNamespace("stats", quietly = TRUE)) {
  i <- 2
  x <- c(le_w[i], le_f[i])
  w <- c(wd[i], fd(wd[i]))
  lubridate::as.duration(stats::weighted.mean(x, w))
}
#> [1] "10157.1428571429s (~2.82 hours)" # Expected

## Converting the output to `hms`

le_w <- lubridate::dhours(1.25)
le_f <- lubridate::dhours(6.23)
wd <- 3
le_week(le_w, le_f, wd)
#> [1] "14744.5714285714s (~4.1 hours)" # Expected

hms::hms(as.numeric(le_week(le_w, le_f, wd)))
#> 04:05:44.571429 # Expected

## Rounding the output at the seconds level

le_w <- lubridate::dhours(3.4094)
le_f <- lubridate::dhours(6.2345)
wd <- 2
le_week(le_w, le_f, wd)
#> [1] "19538.3828571429s (~5.43 hours)" # Expected

round_time(le_week(le_w, le_f, wd))
#> [1] "19538s (~5.43 hours)" # Expected

```

 micro_mctq

 A fictional μ MCTQ dataset

Description

[Maturing]

A fictional dataset, **for testing and learning purposes**, composed of basic/measurable and computed variables of the Munich ChronoType Questionnaire (MCTQ) micro (μ) version.

This data was created following the guidelines in Ghotbi et al. (2020), in addition to the guidelines in Roenneberg, Wirz-Justice, & Mellow (2003), Roenneberg, Allebrandt, Mellow, & Vetter (2012), Jankowski (2017), and The Worldwide Experimental Platform (n.d.). See the References and Details sections to learn more.

Usage

micro_mctq

Format

A [tibble](#) with 19 columns and 50 rows:

id A unique [integer](#) value to identify each respondent in the dataset.

Type: Control.

R class: [integer](#).

shift_work A [logical](#) value indicating if the respondent has been a shift- or night-worker in the past three

wd Number of **workdays** per week.

Statement (EN): "Normally, I work ___ days/week".

Type: Basic.

R class: [integer](#).

fd Number of **work-free days** per week.

Type: Computed.

R class: [integer](#).

so_w Local time of sleep onset on **workdays**.

Statement (EN): "On WORKDAYS ... I normally fall asleep at ___ : ___ AM/PM (this is NOT when you get into bed, but rather when you fall asleep)".

Type: Basic.

R class: [hms](#).

se_w Local time of sleep end on **workdays**.

Statement (EN): "On WORKDAYS ... I normally wake up at ___ : ___ AM/PM (this is NOT when you get out of bed, but rather when you wake up)".

Type: Basic.

R class: [hms](#).

sd_w Sleep duration on **workdays**.

Type: Computed.

R class: [Duration](#).

msw Local time of mid-sleep on **workdays**.

Type: Computed.

R class: [hms](#).

so_f Local time of sleep onset on **work-free days** when the respondent **doesn't** use an alarm clock to wake up.

Statement (EN): "On WORK-FREE DAYS when I DON'T use an alarm clock ... I normally fall asleep at ___ : ___ AM/PM (this is NOT when you get into bed, but rather when you fall asleep)".

Type: Basic.

R class: [hms](#).

se_f Local time of sleep end on **work-free days** when the respondent **doesn't** use an alarm clock to wake up.

Statement (EN): "On WORK-FREE DAYS when I DON'T use an alarm clock ... I normally wake up at ___ : ___ AM/PM (this is NOT when you get out of bed, but rather when you wake up)".

Type: Basic.

R class: [hms](#).

sd_f Sleep duration on **work-free days** when the respondent **doesn't** use an alarm clock to wake up.

Type: Computed.

R class: [Duration](#).

msf Local time of mid-sleep on **work-free days** when the respondent **doesn't** use an alarm clock to wake up.

Type: Computed.

R class: [hms](#).

sd_week Average weekly sleep duration.

Type: Computed.

R class: [Duration](#).

sloss_week Weekly sleep loss.

Type: Computed.

R class: [Duration](#).

msf_sc Chronotype or corrected local time of mid-sleep on **work-free days**.

Type: Computed.

R class: [hms](#).

sjl_rel Relative social jetlag.

Type: Computed.

R class: [Duration](#).

sjl Absolute social jetlag.

Type: Computed.

R class: [Duration](#).

sjl_sc_rel Jankowski's relative sleep-corrected social jetlag.

Type: Computed.

R class: [Duration](#).

sjl_sc Jankowski's sleep-corrected social jetlag.

Type: Computed.

R class: [Duration](#).

Details

micro_mctq is a tidied, validated, and transformed version of `raw_data("micro_mctq.csv")`.

Guidelines:

To learn more about the Munich ChronoType Questionnaire (MCTQ), see Roenneberg, Wirz-Justice, & Mewes (2003), Roenneberg, Allebrandt, Mewes, & Vetter (2012), Roenneberg et al. (2015), and Roenneberg, Pilz, Zerbini, & Winnebeck (2019).

To know about different MCTQ versions, see Juda, Vetter, & Roenneberg (2013) and Ghotbi et al. (2020).

To learn about the sleep-corrected social jetlag, see Jankowski (2017).

If you're curious about the variable computations and want to have access to the full questionnaire, see The Worldwide Experimental Platform (n.d.).

Data building and data wrangling:

This dataset was created by randomized sampling (see `random_mctq()`) and by manual insertions of special cases. Its purpose is to demonstrate common cases and data issues that researchers may find in their MCTQ data, in addition to be a suggested data structure for MCTQ data.

You can see the `micro_mctq` build and data wrangling processes [here](#).

Variable naming:

The naming of the variables took into account the naming scheme used in MCTQ publications, in addition to the guidelines of the [tidyverse style guide](#).

Variable classes:

The `mctq` package works with a set of object classes specially created to hold time values. These classes can be found in the [hms](#) and [lubridate](#) package.

Duration objects:

If you prefer to view `Duration` objects as `hms` objects, run `pretty_mctq(micro_mctq)`.

Source

Created by Daniel Vartanian (package author).

References

- Ghotbi, N., Pilz, L. K., Winnebeck, E. C., Vetter, C., Zerbini, G., Lenssen, D., Frighetto, G., Salamanca, M., Costa, R., Montagnese, S., & Roenneberg, T. (2020). The μ MCTQ: an ultra-short version of the Munich ChronoType Questionnaire. *Journal of Biological Rhythms*, 35(1), 98-110. doi: [10.1177/0748730419886986](https://doi.org/10.1177/0748730419886986).
- Jankowski K. S. (2017). Social jet lag: sleep-corrected formula. *Chronobiology International*, 34(4), 531-535. doi: [10.1080/07420528.2017.1299162](https://doi.org/10.1080/07420528.2017.1299162).
- Juda, M., Vetter, C., & Roenneberg, T. (2013). The Munich ChronoType Questionnaire for shift-workers (MCTQ^{Shift}). *Journal of Biological Rhythms*, 28(2), 130-140. doi: [10.1177/0748730412475041](https://doi.org/10.1177/0748730412475041).
- Roenneberg T., Allebrandt K. V., Mewes M., & Vetter C. (2012). Social jetlag and obesity. *Current Biology*, 22(10), 939-43. doi: [10.1016/j.cub.2012.03.038](https://doi.org/10.1016/j.cub.2012.03.038).
- Roenneberg, T., Keller, L. K., Fischer, D., Matera, J. L., Vetter, C., & Winnebeck, E. C. (2015). Human activity and rest in situ. In A. Sehgal (Ed.), *Methods in Enzymology* (Vol. 552, pp. 257-283). London, UK: Academic Press. doi: [10.1016/bs.mie.2014.11.028](https://doi.org/10.1016/bs.mie.2014.11.028).
- Roenneberg, T., Pilz, L. K., Zerbini, G., & Winnebeck, E. C. (2019). Chronotype and social jetlag: a (self-) critical review. *Biology*, 8(3), 54. doi: [10.3390/biology8030054](https://doi.org/10.3390/biology8030054).

Roenneberg, T., Wirz-Justice, A., & Merrow, M. (2003). Life between clocks: daily temporal patterns of human chronotypes. *Journal of Biological Rhythms*, 18(1), 80-90. doi: [10.1177/0748730402239679](https://doi.org/10.1177/0748730402239679).

The Worldwide Experimental Platform (n.d.). MCTQ. <https://www.thewep.org/documentations/mctq/>

See Also

Other datasets: [shift_mctq](#), [std_mctq](#)

msf_sc	<i>Compute MCTQ sleep-corrected local time of mid-sleep on work-free days</i>
--------	---

Description

[Maturing]

msf_sc() computes the **chronotype or sleep-corrected local time of mid-sleep on work-free days** for standard, micro, and shift versions of the Munich ChronoType Questionnaire (MCTQ).

chronotype() is just a wrapper for msf_sc().

When using the shift version of the MCTQ, replace the value of sd_week to sd_overall, as instructed in the Arguments section.

Usage

```
msf_sc(msf, sd_w, sd_f, sd_week, alarm_f)
```

```
chronotype(msf, sd_w, sd_f, sd_week, alarm_f)
```

Arguments

msf	An hms object corresponding to the local time of mid-sleep on work-free days from a standard, micro, or shift version of the MCTQ questionnaire. You can use msl() to compute it.
sd_w	A Duration object corresponding to the sleep duration on work days from a standard, micro, or shift version of the MCTQ questionnaire. You can use sdu() to compute it.
sd_f	A Duration object corresponding to the sleep duration on work-free days from a standard, micro, or shift version of the MCTQ questionnaire. You can use sdu() to compute it.
sd_week	A Duration object corresponding to the average weekly sleep duration from a standard or micro version of the MCTQ questionnaire (you can use sd_week() to compute it) or the overall sleep duration of a particular shift from a shift version of the MCTQ questionnaire (you can use sd_overall() to compute it).

`alarm_f` A **logical** object corresponding to the **alarm clock use on work-free days** from a standard, micro, or shift version of the MCTQ questionnaire. Note that, if `alarm_f == TRUE`, `msf_sc` cannot be computed, `msf_sc()` will return NA for these cases. For the μ MCTQ, this value must be set as FALSE all times, since the questionnaire considers only the work-free days when the respondent does not use an alarm (e.g., `alarm_f = rep(FALSE, length(msf))`).

Details

Standard MCTQ functions were created following the guidelines in Roenneberg, Wirz-Justice, & Merrow (2003), Roenneberg, Allebrandt, Merrow, & Vetter (2012), and from The Worldwide Experimental Platform (theWeP, n.d.).

μ **MCTQ** functions were created following the guidelines in Ghotbi et al. (2020), in addition to the guidelines used for the standard MCTQ.

MCTQ^{Shift} functions were created following the guidelines in Juda, Vetter, & Roenneberg (2013), in addition to the guidelines used for the standard MCTQ.

See the References section to learn more.

Class requirements:

The `mctq` package works with a set of object classes specially created to hold time values. These classes can be found in the `lubridate` and `hms` packages. Please refer to those package documents to learn more about them.

Rounding and fractional time:

Some operations may produce an output with fractional time (e.g., "19538.3828571429s (~5.43 hours)", 01:15:44.505). If you want, you can round it with `round_time()`.

Our recommendation is to avoid rounding, but, if you do, make sure that you only round your values after all computations are done. That way you avoid **round-off errors**.

Value

An `hms` object corresponding to the MCTQ chronotype or sleep-corrected local time of mid-sleep on work-free days.

Guidelines

Roenneberg, Allebrandt, Merrow, & Vetter (2012), Ghotbi et al. (2020), Juda, Vetter, & Roenneberg (2013), and The Worldwide Experimental Platform (n.d.) guidelines for `msf_sc()` (MSF_{sc}) computation are as follows.

Notes:

- For all cases, MSF_{sc} cannot be computed if the participant wakes up with an alarm clock on work-free days ($Alarm_F$).
- For MCTQ^{Shift}, the computation below must be applied to each shift section of the questionnaire.
- MSF_{sc} is a proxy for the subject chronotype in standard and micro versions of the MCTQ.

- The basis for estimating chronotype in shift-workers is the mid-sleep on work-free days after evening shifts (MSF^E). In case work schedules do not comprise evening shifts, Juda, Vetter, & Roenneberg (2013) propose to derive it from the MSF_{sc} of other shifts (e.g., by using a linear model). Unfortunately, the `mctq` package can't help you with that, as it requires a closer look at your data.
- MSF_{sc} depends on developmental and environmental conditions (e.g., age, light exposure). For epidemiological and genetic studies, MSF_{sc} must be normalized for age and sex to make populations of different age and sex compositions comparable (Roenneberg, Allebrandt, Merrow, & Vetter, 2012).
- If you are visualizing this documentation in plain text, you may have some trouble understanding the equations. You can see this documentation on the package [website](#).

For standard and micro versions of the MCTQ:

$$\text{If } SD_F \leq SD_W, MSF$$

$$\text{If } SD_F > SD_W, MSF - \frac{SD_F - SD_{week}}{2}$$

Where:

- MSF = Local time of mid-sleep on work-free days.
- SD_W = Sleep duration on workdays.
- SD_F = Sleep duration on work-free days.
- SD_{week} = Average weekly sleep duration.

* W = Workdays; F = Work-free days.

For the shift version of the MCTQ:

$$\text{If } SD_F^{M/E/N} \leq SD_W^{M/E/N}, MSF^{M/E/N}$$

$$\text{If } SD_F^{M/E/N} > SD_W^{M/E/N}, MSF^{M/E/N} - \frac{SD_F^{M/E/N} - \emptyset SD^{M/E/N}}{2}$$

Where:

- $MSF^{M/E/N}$ = Local time of mid-sleep between two free days after a particular shift.
- $SD_W^{M/E/N}$ = Sleep duration between two days in a particular shift.
- $SD_F^{M/E/N}$ = Sleep duration between two free days after a particular shift.
- $\emptyset SD^{M/E/N}$ = Overall sleep duration of a particular shift.

* W = Workdays; F = Work-free days, M = Morning shift; E = Evening shift; N = Night shift.

References

- Ghotbi, N., Pilz, L. K., Winnebeck, E. C., Vetter, C., Zerbini, G., Lenssen, D., Frighetto, G., Salamanca, M., Costa, R., Montagnese, S., & Roenneberg, T. (2020). The μ MCTQ: an ultra-short version of the Munich ChronoType Questionnaire. *Journal of Biological Rhythms*, 35(1), 98-110. doi: [10.1177/0748730419886986](https://doi.org/10.1177/0748730419886986).
- Juda, M., Vetter, C., & Roenneberg, T. (2013). The Munich ChronoType Questionnaire for shift-workers (MCTQ^{Shift}). *Journal of Biological Rhythms*, 28(2), 130-140. doi: [10.1177/0748730412475041](https://doi.org/10.1177/0748730412475041).
- Roenneberg T., Allebrandt K. V., Mellow M., & Vetter C. (2012). Social jetlag and obesity. *Current Biology*, 22(10), 939-43. doi: [10.1016/j.cub.2012.03.038](https://doi.org/10.1016/j.cub.2012.03.038).
- Roenneberg, T., Wirz-Justice, A., & Mellow, M. (2003). Life between clocks: daily temporal patterns of human chronotypes. *Journal of Biological Rhythms*, 18(1), 80-90. doi: [10.1177/0748730402239679](https://doi.org/10.1177/0748730402239679).
- The Worldwide Experimental Platform (n.d.). MCTQ. <https://www.thewep.org/documentations/mctq/>

See Also

Other MCTQ functions: [fd\(\)](#), [gu\(\)](#), [le_week\(\)](#), [msl\(\)](#), [napd\(\)](#), [sd24\(\)](#), [sd_overall\(\)](#), [sd_week\(\)](#), [sdu\(\)](#), [sjl_sc\(\)](#), [sjl_weighted\(\)](#), [sjl\(\)](#), [so\(\)](#), [tbt\(\)](#)

Examples

```
## Scalar example

msf <- hms::parse_hms("04:00:00")
sd_w <- lubridate::dhours(6)
sd_f <- lubridate::dhours(7)
sd_week <- lubridate::dhours(6.29)
alarm_f <- FALSE
msf_sc(msf, sd_w, sd_f, sd_week, alarm_f)
#> 03:38:42 # Expected

msf <- hms::parse_hm("01:00:00")
sd_w <- lubridate::dhours(5.5)
sd_f <- lubridate::dhours(9)
sd_week <- lubridate::dhours(6.75)
alarm_f <- FALSE
msf_sc(msf, sd_w, sd_f, sd_week, alarm_f)
#> 23:52:30 # Expected

msf <- hms::parse_hms("05:40:00")
sd_w <- lubridate::dhours(7.5)
sd_f <- lubridate::dhours(10)
sd_week <- lubridate::dhours(8.5)
alarm_f <- TRUE
msf_sc(msf, sd_w, sd_f, sd_week, alarm_f)
#> NA # Expected (`msf_sc` cannot be computed if `alarm_f == TRUE`)
```

```
## Vector example
```

```

msf <- c(hms::parse_hms("03:45:00"), hms::parse_hm("04:45:00"))
sd_w <- c(lubridate::dhours(9), lubridate::dhours(6.45))
sd_f <- c(lubridate::dhours(5), lubridate::dhours(10))
sd_week <- c(lubridate::dhours(8.5), lubridate::dhours(9.2))
alarm_f <- c(FALSE, FALSE)
msf_sc(msf, sd_w, sd_f, sd_week, alarm_f)
#> 03:45:00 # Expected
#> 04:21:00 # Expected

## chronotype(): A wrapper for msf_sc()

msf <- hms::parse_hms("07:00:00")
sd_w <- lubridate::dhours(6)
sd_f <- lubridate::dhours(12)
sd_week <- lubridate::dhours(9.45)
alarm_f <- FALSE
chronotype(msf, sd_w, sd_f, sd_week, alarm_f)
#> 05:43:30 # Expected

## Rounding the output at the seconds level

msf <- hms::parse_hms("05:40:00")
sd_w <- lubridate::dhours(5.43678)
sd_f <- lubridate::dhours(9.345111)
sd_week <- lubridate::dhours(7.5453)
alarm_f <- FALSE
msf_sc(msf, sd_w, sd_f, sd_week, alarm_f)
#> 04:46:00.3402 # Expected

round_time(msf_sc(msf, sd_w, sd_f, sd_week, alarm_f))
#> 04:46:00 # Expected

```

msl

Compute MCTQ local time of mid-sleep

Description

[Maturing]

`msl()` computes the **local time of mid-sleep** for standard, micro, and shift versions of the Munich ChronoType Questionnaire (MCTQ).

Please note that, although we tried to preserve the original authors' naming pattern for the MCTQ functions, the name `ms` provokes a dangerous name collision with the `ms()` function (a function for parsing minutes and seconds components). That's why we named it `msl`. `msl()` and `sdu()` are the only exceptions, all the other `mctq` functions maintain a strong naming resemblance with the original authors' naming pattern.

Usage

```
msl(so, sd)
```


Arguments

so	An hms object corresponding to the local time of sleep onset from a standard, micro, or shift version of the MCTQ questionnaire. You can use so() to compute it for the standard or shift version.
sd	A Duration object corresponding to the sleep duration from a standard, micro, or shift version of the MCTQ questionnaire. You can use sdu() to compute it for any MCTQ version.

Details

Standard MCTQ functions were created following the guidelines in Roenneberg, Wirz-Justice, & Meroow (2003), Roenneberg, Allebrandt, Meroow, & Vetter (2012), and from The Worldwide Experimental Platform (theWeP, n.d.).

μ **MCTQ** functions were created following the guidelines in Ghotbi et al. (2020), in addition to the guidelines used for the standard MCTQ.

MCTQ^{Shift} functions were created following the guidelines in Juda, Vetter, & Roenneberg (2013), in addition to the guidelines used for the standard MCTQ.

See the References section to learn more.

Class requirements:

The `mctq` package works with a set of object classes specially created to hold time values. These classes can be found in the [lubridate](#) and [hms](#) packages. Please refer to those package documents to learn more about them.

Rounding and fractional time:

Some operations may produce an output with fractional time (e.g., "19538.3828571429s (~5.43 hours)", 01:15:44.505). If you want, you can round it with [round_time\(\)](#).

Our recommendation is to avoid rounding, but, if you do, make sure that you only round your values after all computations are done. That way you avoid **round-off errors**.

Value

An [hms](#) object corresponding to the vectorized sum of `so` and `(sd / 2)` in a circular time frame of 24 hours.

Guidelines

Roenneberg, Allebrandt, Meroow, & Vetter (2012), Ghotbi et al. (2020), Juda, Vetter, & Roenneberg (2013), and The Worldwide Experimental Platform (n.d.) guidelines for `msl()` (*MSW* or *MSF*) computation are as follows.

Notes:

- This computation must be applied to each section of the questionnaire.
- If you are visualizing this documentation in plain text, you may have some trouble understanding the equations. You can see this documentation on the package [website](#).

For standard and micro versions of the MCTQ:

$$SO_{W/F} + \frac{SD_{W/F}}{2}$$

Where:

- $SO_{W/F}$ = Local time of sleep onset on work **or** work-free days.
- $SD_{W/F}$ = Sleep duration on work **or** work-free days.

* W = Workdays; F = Work-free days.

For the shift version of the MCTQ:

$$SO_{W/F}^{M/E/N} + \frac{SD_{W/F}^{M/E/N}}{2}$$

Where:

- $SO_{W/F}^{M/E/N}$ = Local time of sleep onset between two days in a particular shift **or** between two free days after a particular shift.
- $SD_{W/F}^{M/E/N}$ = Sleep duration between two days in a particular shift **or** between two free days after a particular shift.

* W = Workdays; F = Work-free days, M = Morning shift; E = Evening shift; N = Night shift.

References

- Ghotbi, N., Pilz, L. K., Winnebeck, E. C., Vetter, C., Zerbini, G., Lenssen, D., Frighetto, G., Salamanca, M., Costa, R., Montagnese, S., & Roenneberg, T. (2020). The μ MCTQ: an ultra-short version of the Munich ChronoType Questionnaire. *Journal of Biological Rhythms*, 35(1), 98-110. doi: [10.1177/0748730419886986](https://doi.org/10.1177/0748730419886986).
- Juda, M., Vetter, C., & Roenneberg, T. (2013). The Munich ChronoType Questionnaire for shift-workers (MCTQ^{Shift}). *Journal of Biological Rhythms*, 28(2), 130-140. doi: [10.1177/0748730412475041](https://doi.org/10.1177/0748730412475041).
- Roenneberg T., Allebrandt K. V., Merrow M., & Vetter C. (2012). Social jetlag and obesity. *Current Biology*, 22(10), 939-43. doi: [10.1016/j.cub.2012.03.038](https://doi.org/10.1016/j.cub.2012.03.038).
- Roenneberg, T., Wirz-Justice, A., & Merrow, M. (2003). Life between clocks: daily temporal patterns of human chronotypes. *Journal of Biological Rhythms*, 18(1), 80-90. doi: [10.1177/0748730402239679](https://doi.org/10.1177/0748730402239679).
- The Worldwide Experimental Platform (n.d.). MCTQ. <https://www.thewep.org/documentations/mctq/>

See Also

Other MCTQ functions: [fd\(\)](#), [gu\(\)](#), [le_week\(\)](#), [msf_sc\(\)](#), [napd\(\)](#), [sd24\(\)](#), [sd_overall\(\)](#), [sd_week\(\)](#), [sdu\(\)](#), [sjl_sc\(\)](#), [sjl_weighted\(\)](#), [sjl\(\)](#), [so\(\)](#), [tbt\(\)](#)

Examples

```
## Scalar example

so <- hms::parse_hm("23:30")
sd <- lubridate::dhours(8)
msl(so, sd)
#> 03:30:00 # Expected

so <- hms::parse_hm("01:00")
sd <- lubridate::dhours(10)
msl(so, sd)
#> 06:00:00 # Expected

so <- hms::as_hms(NA)
sd <- lubridate::dhours(7.5)
msl(so, sd)
#> NA # Expected

## Vector example

so <- c(hms::parse_hm("00:10"), hms::parse_hm("01:15"))
sd <- c(lubridate::dhours(9.25), lubridate::dhours(5.45))
msl(so, sd)
#> [1] 04:47:30 # Expected
#> [1] 03:58:30 # Expected
```

 napd

Compute MCTQ nap duration (only for MCTQ`Shift)

Description**[Maturing]**

napd() computes the **nap duration** for the shift version of the Munich ChronoType Questionnaire (MCTQ).

Usage

```
napd(napo, nape)
```

Arguments

napo	An hms object corresponding to the local time of nap onset from the shift version of the MCTQ questionnaire.
nape	An hms object corresponding to the local time of nap end from the shift version of the MCTQ questionnaire.

Details

Standard MCTQ functions were created following the guidelines in Roenneberg, Wirz-Justice, & Meroow (2003), Roenneberg, Allebrandt, Meroow, & Vetter (2012), and from The Worldwide Experimental Platform (theWeP, n.d.).

μ **MCTQ** functions were created following the guidelines in Ghotbi et al. (2020), in addition to the guidelines used for the standard MCTQ.

MCTQ^{Shift} functions were created following the guidelines in Juda, Vetter, & Roenneberg (2013), in addition to the guidelines used for the standard MCTQ.

See the References section to learn more.

Class requirements:

The `mctq` package works with a set of object classes specially created to hold time values. These classes can be found in the `lubridate` and `hms` packages. Please refer to those package documents to learn more about them.

Rounding and fractional time:

Some operations may produce an output with fractional time (e.g., "19538.3828571429s (~5.43 hours)", "01:15:44.505"). If you want, you can round it with `round_time()`.

Our recommendation is to avoid rounding, but, if you do, make sure that you only round your values after all computations are done. That way you avoid **round-off errors**.

Value

A `Duration` object corresponding to the vectorized difference between `nape` and `napo` in a circular time frame of 24 hours.

Guidelines

Juda, Vetter & Roenneberg (2013) and The Worldwide Experimental Platform (n.d.) guidelines for `napd()` (*NapD*) computation are as follows.

Notes:

- This computation must be applied to each section of the questionnaire.
- If you are visualizing this documentation in plain text, you may have some trouble understanding the equations. You can see this documentation on the package [website](#).

Computation:

$$NapE_{W/F}^{M/E/N} - NapO_{W/F}^{M/E/N}$$

Where:

- $NapO_{W/F}^{M/E/N}$ = Local time of nap onset between two days in a particular shift **or** between two free days after a particular shift ("I take a nap from ___ o'clock [...]").
- $NapE_{W/F}^{M/E/N}$ = Local time of nap end between two days in a particular shift **or** between two free days after a particular shift ("[...] to ___ o'clock").

* W = Workdays; F = Work-free days, M = Morning shift; E = Evening shift; N = Night shift.

References

- Ghotbi, N., Pilz, L. K., Winnebeck, E. C., Vetter, C., Zerbini, G., Lenssen, D., Frighetto, G., Salamanca, M., Costa, R., Montagnese, S., & Roenneberg, T. (2020). The μ MCTQ: an ultra-short version of the Munich ChronoType Questionnaire. *Journal of Biological Rhythms*, 35(1), 98-110. doi: [10.1177/0748730419886986](https://doi.org/10.1177/0748730419886986).
- Juda, M., Vetter, C., & Roenneberg, T. (2013). The Munich ChronoType Questionnaire for shift-workers (MCTQ^{Shift}). *Journal of Biological Rhythms*, 28(2), 130-140. doi: [10.1177/0748730412475041](https://doi.org/10.1177/0748730412475041).
- Roenneberg T., Allebrandt K. V., Merrow M., & Vetter C. (2012). Social jetlag and obesity. *Current Biology*, 22(10), 939-43. doi: [10.1016/j.cub.2012.03.038](https://doi.org/10.1016/j.cub.2012.03.038).
- Roenneberg, T., Wirz-Justice, A., & Merrow, M. (2003). Life between clocks: daily temporal patterns of human chronotypes. *Journal of Biological Rhythms*, 18(1), 80-90. doi: [10.1177/0748730402239679](https://doi.org/10.1177/0748730402239679).
- The Worldwide Experimental Platform (n.d.). MCTQ. <https://www.thewep.org/documentations/mctq/>

See Also

Other MCTQ functions: [fd\(\)](#), [gu\(\)](#), [le_week\(\)](#), [msf_sc\(\)](#), [msl\(\)](#), [sd24\(\)](#), [sd_overall\(\)](#), [sd_week\(\)](#), [sdu\(\)](#), [sjl_sc\(\)](#), [sjl_weighted\(\)](#), [sjl\(\)](#), [so\(\)](#), [tbt\(\)](#)

Examples

```
## Scalar example

napo <- hms::parse_hm("12:30")
nape <- hms::parse_hm("14:20")
napd(napo, nape)
#> [1] "6600s (~1.83 hours)" # Expected

napo <- hms::parse_hm("23:45")
nape <- hms::parse_hm("00:30")
napd(napo, nape)
#> [1] "2700s (~45 minutes)" # Expected

napo <- hms::parse_hm("10:20")
nape <- hms::as_hms(NA)
napd(napo, nape)
#> [1] NA # Expected

## Vector example

napo <- c(hms::parse_hm("01:25"), hms::parse_hm("23:50"))
nape <- c(hms::parse_hm("03:10"), hms::parse_hm("01:10"))
napd(napo, nape)
#> [1] "6300s (~1.75 hours)" "4800s (~1.33 hours)" # Expected
```

`pretty_mctq`*Make an MCTQ dataset more presentable*

Description

[Maturing]

`pretty_mctq()` helps you to transform your Munich ChronoType Questionnaire (MCTQ) data in many ways. See the Arguments and Details section to learn more.

Usage

```
pretty_mctq(data, round = TRUE, hms = TRUE)
```

Arguments

<code>data</code>	A <code>data.frame</code> object.
<code>round</code>	(optional) a <code>logical</code> value indicating if <code>Duration</code> and <code>hms</code> objects must be rounded at the seconds level (default: <code>TRUE</code>).
<code>hms</code>	(optional) a <code>logical</code> value indicating if <code>Duration</code> and <code>difftime</code> objects must be converted to <code>hms</code> (default: <code>TRUE</code>).

Details

Rounding:

Please note that by rounding MCTQ values you discard data. That is to say that if you need to redo a computation, or do new ones, your values can be off by a couple of seconds (see **round-off error**).

Round your values only if and when you want to present them more clearly, like in graphical representations. You can also round values to facilitate data exporting to text formats (like `.csv`), but note that this will come with a precision cost.

Note also that `pretty_mctq()` uses `round_time()` for rounding. `round_time()` is based on `round()`, which uses the IEC 60559 standard. For more information see the `round_time()` documentation.

Value

A transformed `data.frame` object, as indicated in the arguments.

See Also

Other utility functions: `assign_date()`, `cycle_time()`, `qplot_walk()`, `random_mctq()`, `raw_data()`, `round_time()`, `shorter_interval()`, `sum_time()`

Examples

```

data <- data.frame(
  a = 1,
  b = lubridate::duration(1.12345),
  c = hms::hms(1.12345)
)

## Rounding time objects from `data`

pretty_mctq(data, round = TRUE, hms = FALSE)

## Converting non-'hms' time objects from 'data' to 'hms'

pretty_mctq(data, round = FALSE, hms = TRUE)

```

qplot_walk

*Walk through distribution plots***Description****[Maturing]**

qplot_walk() helps you to visually assess the distribution of your data. It uses `qplot()` to walk through each selected variable from a data frame.

Usage

```

qplot_walk(
  data,
  ...,
  cols = NULL,
  pattern = NULL,
  ignore = "character",
  remove_id = TRUE,
  midday_change = TRUE
)

```

Arguments

data	An atomic or a data.frame object.
...	(optional) additional arguments to be passed to <code>qplot()</code> .
cols	(optional) (only for data frames) a character object indicating column names in data for plotting. If NULL, <code>qplot_walk()</code> will use all columns in data. This setting only works if <code>pattern = NULL</code> (default: NULL).
pattern	(optional) (only for data frames) a string with a regular expression to select column names in data for plotting. This setting only works if <code>cols = NULL</code> (default: NULL).

ignore	(optional) (only for data frames) a character object indicating which object classes the function must ignore. This setting can be used with <code>cols</code> and <code>pattern</code> . Assign <code>NULL</code> to disable this behavior (default: "character").
remove_id	(optional) (only for data frames) a logical value indicating if the function must ignore column names in data that match with the regular expression <code>"^id\$ [\._-\]id\$"</code> (default: <code>TRUE</code>).
midday_change	(optional) a logical value indicating if the function must apply a midday change for hms variables with values greater than <code>22:00:00</code> (see the Details section to learn more) (default: <code>TRUE</code>).

Details

Requirements:

This function requires the [utils](#), [grDevices](#), and [ggplot2](#) packages and can only run in interactive mode. The [utils](#) and [grDevices](#) packages comes with a standard R installation and is typically loaded by default. Most people also run R interactively.

If you don't have any or one of the packages mentioned above, you can install them with `install.packages("utils")`, `install.packages("grDevices")`, and `install.packages("ggplot2")`.

Plot recover:

`qplot_walk()` clears all plots after it runs. For that reason, the function first emits a dialog message warning the user of this behavior before it runs. If you want to recover a single distribution plot, assign the variable vector to the data argument.

Additional arguments to `qplot()`:

`qplot_walk()` uses [ggplot2](#) `qplot()` to generate plots. If you are familiar with `qplot()`, you can pass additional arguments to the function using the ellipsis argument (`...`).

Note that `x`, `y`, and `data` arguments are reserved for `qplot_walk()`.

Duration, Period, and `difftime` objects:

To help with the visualization, `qplot_walk()` automatically converts [Duration](#), [Period](#), and [difftime](#) objects to [hms](#).

Midday change:

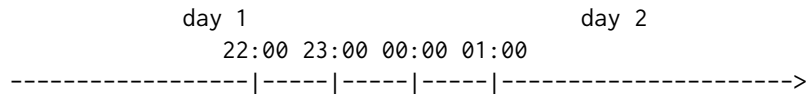
Time variables with values greater than `22:00:00` will automatically be converted to [POSIXct](#) and be attached to a two-day timeline using the midday hour as a cutting point, i.e., all values with 12 hours or more will be placed on day 1, and all the rest will be placed on day 2.

This is made to better represent time vectors that cross the midnight hour. You can disable this behavior by using `midday_change = FALSE`.

Example: Say you have a vector of time values that cross the midnight hour (e.g., an [hms](#) vector with `22:00`, `23:00`, `00:00`, `01:00` values). If you use `midday_change = FALSE`, your data will be represented linearly.

```
00:00 01:00                                22:00 23:00
|-----|-----|-----|-----|-----|-----|-----|-----|----->
```

By using `midday_change = TRUE` (default), `qplot_walk()` will fit your data to a circular time frame of 24 hours.

**id variables:**

qplot_walk() will ignore any variable with the follow name pattern "`^id$|[\._-\]id$`", i.e., any variable named `id` or that ends with `.id`, `_id`, or `-id`.

You can disable this behavior using `remove_id = FALSE`.

Value

An invisible NULL. This function don't aim to return values.

See Also

Other utility functions: [assign_date\(\)](#), [cycle_time\(\)](#), [pretty_mctq\(\)](#), [random_mctq\(\)](#), [raw_data\(\)](#), [round_time\(\)](#), [shorter_interval\(\)](#), [sum_time\(\)](#)

Examples

```
if (interactive()) {
  ## Ploting a single column from 'data'
  qplot_walk(mctq::std_mctq$bt_w)
  ## Ploting all columns from 'data'
  qplot_walk(mctq::std_mctq, ignore = NULL, remove_id = FALSE)
  ## Ploting selected columns from 'data'
  qplot_walk(mctq::std_mctq, cols = c("bt_w", "msf_sc"))
  ## Ploting selected columns from 'data' using a name pattern
  qplot_walk(mctq::std_mctq, pattern = "_w$")
  ## Examples using other datasets
  if (requireNamespace("datasets", quietly = TRUE)) {
    qplot_walk(datasets::iris)
  }
}
```

`random_mctq`*Build a random MCTQ case*

Description

[Maturing]

`random_mctq` builds a fictional Munich ChronoType Questionnaire (MCTQ) case composed of MCTQ basic/measurable variables.

This function is **for testing and learning purposes only**. Please don't misuse it.

Usage

```
random_mctq(model = "standard")
```

Arguments

`model` A string indicating the data model to return. Valid values are: "standard", "shift", and "micro" (default: "standard").

Details

The case structure (variable names and classes) are the same as the datasets provided by the `mctq` package. See `?std_mctq`, `?micro_mctq` and `?shift_mctq` to learn more.

Requirements:

This function requires the `stats` package. This won't be an issue for most people since the package comes with a standard R installation.

If you don't have the `stats` package, you can install it with `install.packages("stats")`.

Cases:

Random standard and micro MCTQ cases were created with the general population in mind. The data was set to resemble the distribution parameters shown in Roenneberg, Wirz-Justice, & Merrow (2003).

MCTQ^{Shift} random cases were created based on the shift configuration from "Study Site 1" shown in Vetter, Juda, & Roenneberg (2012). The data was set to resemble the distribution parameters shown in Juda, Vetter, & Roenneberg (2013).

You can see more about the distribution parameters used [here](#).

Value

A named `list` with elements representing each MCTQ basic/measurable variable of the model indicated in the `model` argument.

References

Juda, M., Vetter, C., & Roenneberg, T. (2013). The Munich ChronoType Questionnaire for shiftworkers (MCTQ^{Shift}). *Journal of Biological Rhythms*, 28(2), 130-140. doi: [10.1177/0748730412475041](https://doi.org/10.1177/0748730412475041).

Roenneberg, T., Wirz-Justice, A., & Mewes, M. (2003). Life between clocks: daily temporal patterns of human chronotypes. *Journal of Biological Rhythms*, 18(1), 80-90. doi: [10.1177/0748730402239679](https://doi.org/10.1177/0748730402239679).

Vetter, C., Juda, M., & Roenneberg, T. (2012). The influence of internal time, time awake, and sleep duration on cognitive performance in shiftworkers. *Chronobiology International*, 29(8), 1127-1138. doi: [10.3109/07420528.2012.707999](https://doi.org/10.3109/07420528.2012.707999).

See Also

Other utility functions: [assign_date\(\)](#), [cycle_time\(\)](#), [pretty_mctq\(\)](#), [qplot_walk\(\)](#), [raw_data\(\)](#), [round_time\(\)](#), [shorter_interval\(\)](#), [sum_time\(\)](#)

Examples

```
## Not run:
random_mctq("standard")
random_mctq("micro")
random_mctq("shift")

## End(Not run)
```

raw_data

Get paths to mctq raw datasets

Description

[Maturing]

mctq comes bundled with raw fictional datasets for testing and learning purposes. `raw_data()` makes it easy to access their paths.

Usage

```
raw_data(file = NULL)
```

Arguments

`file` (optional) a [character](#) object indicating the raw data file name(s). If NULL, all raw data file names will be returned (default: NULL).

Value

If `file == NULL`, a [character](#) object with all file names available. Else, a string with the file name path.

See Also

Other utility functions: [assign_date\(\)](#), [cycle_time\(\)](#), [pretty_mctq\(\)](#), [qplot_walk\(\)](#), [random_mctq\(\)](#), [round_time\(\)](#), [shorter_interval\(\)](#), [sum_time\(\)](#)

Examples

```
## Not run:
## To list all raw data file names available

raw_data()

## To get the file path from a specific raw data

raw_data("std_mctq.csv")
## End(Not run)
```

round_time	<i>Round time objects</i>
------------	---------------------------

Description**[Maturing]**

`round_time()` takes a [Duration](#), [difftime](#), [hms](#), [POSIXct](#), or [POSIXlt](#) object and round it at the seconds level.

Usage

```
round_time(x)

## S3 method for class 'Duration'
round_time(x)

## S3 method for class 'difftime'
round_time(x)

## S3 method for class 'hms'
round_time(x)

## S3 method for class 'POSIXct'
round_time(x)

## S3 method for class 'POSIXlt'
round_time(x)
```

Arguments

x An object belonging to one of the following classes: [Duration](#), [difftime](#), [hms](#), [POSIXct](#), or [POSIXlt](#).

Details

Round standard:

round_time() uses `base::round()` for rounding. That is to say that round_time() uses the same IEC 60559 standard ("go to the even digit") for rounding off a 5. Therefore, round(0.5) is equal to 0 and round(-1.5) is equal to -2. See [?round](#) to learn more.

Period objects:

[Period](#) objects are special type of objects developed by the [lubridate](#) team that represents "human units", ignoring possible timeline irregularities. That is to say that 1 day as Period can have different time spans, when looking to a timeline after a irregularity event.

Since the time span of a [Period](#) object can fluctuate, round_time() don't accept this kind of object. You can transform it to a [Duration](#) object and still use the function, but beware that this can produce errors.

Learn more about [Period](#) objects in the [Dates and times](#) chapter of Wickham & Grolemund book (n.d.).

Value

An object of the same class of x rounded at the seconds level.

References

Wickham, H., & Grolemund, G. (n.d.). *R for data science*. Sebastopol, CA: O'Reilly Media.
<https://r4ds.had.co.nz>

See Also

Other date-time rounding functions: [round_hms\(\)](#) [trunc_hms\(\)](#) [round_date\(\)](#).

Other utility functions: [assign_date\(\)](#), [cycle_time\(\)](#), [pretty_mctq\(\)](#), [qplot_walk\(\)](#), [random_mctq\(\)](#), [raw_data\(\)](#), [shorter_interval\(\)](#), [sum_time\(\)](#)

Examples

```
## Scalar example

lubridate::dmilliseconds(123456789)
#> [1] "123456.789s (~1.43 days)" # Expected
round_time(lubridate::dmilliseconds(123456789))
#> [1] "123457s (~1.43 days)" # Expected

as.difftime(12345.6789, units = "secs")
#> Time difference of 12345.68 secs # Expected
round_time(as.difftime(12345.6789, units = "secs"))
#> Time difference of 12346 secs # Expected

hms::as_hms(12345.6789)
#> 03:25:45.6789 # Expected
round_time(hms::as_hms(12345.6789))
#> 03:25:46 # Expected
```

```
lubridate::as_datetime(12345.6789, tz = "EST")
#> [1] "1969-12-31 22:25:45 EST" # Expected
as.numeric(lubridate::as_datetime(12345.6789, tz = "EST"))
#> [1] 12345.68 # Expected
round_time(lubridate::as_datetime(12345.6789, tz = "EST"))
#> [1] "1969-12-31 22:25:46 EST" # Expected
as.numeric(round_time(lubridate::as_datetime(12345.6789, tz = "EST")))
#> [1] 12346 # Expected

## Vector example

c(lubridate::dhours(5.6987), lubridate::dhours(2.6875154))
#> [1] "20515.32s (~5.7 hours)" "9675.05544s (~2.69 hours)" # Expected
round_time(c(lubridate::dhours(5.6987), lubridate::dhours(2.6875154)))
#> [1] "20515s (~5.7 hours)" "9675s (~2.69 hours)" # Expected
```

sd24

Compute MCTQ 24 hours sleep duration (only for MCTQ^{Shift})

Description

[Maturing]

sd24() computes the **24 hours sleep duration** for the shift version of the Munich ChronoType Questionnaire (MCTQ).

Usage

```
sd24(sd, napd, nap)
```

Arguments

sd	A Duration object corresponding to the sleep duration from the shift version of the MCTQ questionnaire. You can use sdu() to compute it.
napd	A Duration object corresponding to the nap duration from the shift version of the MCTQ questionnaire. You can use napd() to compute it.
nap	A logical value corresponding to the "I usually take a nap" response from the shift version of the MCTQ questionnaire.

Details

Standard MCTQ functions were created following the guidelines in Roenneberg, Wirz-Justice, & Mellow (2003), Roenneberg, Allebrandt, Mellow, & Vetter (2012), and from The Worldwide Experimental Platform (theWeP, n.d.).

μ MCTQ functions were created following the guidelines in Ghotbi et al. (2020), in addition to the guidelines used for the standard MCTQ.

MCTQ^{Shift} functions were created following the guidelines in Juda, Vetter, & Roenneberg (2013), in addition to the guidelines used for the standard MCTQ.

See the References section to learn more.

Class requirements:

The `mctq` package works with a set of object classes specially created to hold time values. These classes can be found in the `lubridate` and `hms` packages. Please refer to those package documentations to learn more about them.

Rounding and fractional time:

Some operations may produce an output with fractional time (e.g., "19538.3828571429s (~5.43 hours)", 01:15:44.505). If you want, you can round it with `round_time()`.

Our recommendation is to avoid rounding, but, if you do, make sure that you only round your values after all computations are done. That way you avoid **round-off errors**.

Value

- If `nap == TRUE`, a `Duration` object corresponding to the vectorized sum of `sd` and `napd` in a circular time frame of 24 hours.
- If `nap == FALSE`, a `Duration` object equal to `sd`.

Guidelines

Juda, Vetter & Roenneberg (2013) and The Worldwide Experimental Platform (n.d.) guidelines for `sd24()` (SD_{24}) computation are as follows.

Notes:

- This computation must be applied to each section of the questionnaire.
- If the respondent don't usually take a nap in a particular shift **or** between two free days after a particular shift, `sd24()` will return only $SD_{W/F}^{M/E/N}$.
- If you are visualizing this documentation in plain text, you may have some trouble understanding the equations. You can see this documentation on the package [website](#).

Computation:

$$SD_{W/F}^{M/E/N} + NapD_{W/F}^{M/E/N}$$

Where:

- $SD_{W/F}^{M/E/N}$ = Sleep duration between two days in a particular shift **or** between two free days after a particular shift.
- $NapD_{W/F}^{M/E/N}$ = Nap duration between two days in a particular shift **or** between two free days after a particular shift.

* W = Workdays; F = Work-free days, M = Morning shift; E = Evening shift; N = Night shift.

References

- Ghotbi, N., Pilz, L. K., Winnebeck, E. C., Vetter, C., Zerbini, G., Lenssen, D., Frighetto, G., Salamanca, M., Costa, R., Montagnese, S., & Roenneberg, T. (2020). The μ MCTQ: an ultra-short version of the Munich ChronoType Questionnaire. *Journal of Biological Rhythms*, 35(1), 98-110. doi: [10.1177/0748730419886986](https://doi.org/10.1177/0748730419886986).
- Juda, M., Vetter, C., & Roenneberg, T. (2013). The Munich ChronoType Questionnaire for shift-workers (MCTQ^{Shift}). *Journal of Biological Rhythms*, 28(2), 130-140. doi: [10.1177/0748730412475041](https://doi.org/10.1177/0748730412475041).
- Roenneberg T., Allebrandt K. V., Mellow M., & Vetter C. (2012). Social jetlag and obesity. *Current Biology*, 22(10), 939-43. doi: [10.1016/j.cub.2012.03.038](https://doi.org/10.1016/j.cub.2012.03.038).
- Roenneberg, T., Wirz-Justice, A., & Mellow, M. (2003). Life between clocks: daily temporal patterns of human chronotypes. *Journal of Biological Rhythms*, 18(1), 80-90. doi: [10.1177/0748730402239679](https://doi.org/10.1177/0748730402239679).
- The Worldwide Experimental Platform (n.d.). MCTQ. <https://www.thewep.org/documentations/mctq/>

See Also

Other MCTQ functions: [fd\(\)](#), [gu\(\)](#), [le_week\(\)](#), [msf_sc\(\)](#), [msl\(\)](#), [napd\(\)](#), [sd_overall\(\)](#), [sd_week\(\)](#), [sdu\(\)](#), [sjl_sc\(\)](#), [sjl_weighted\(\)](#), [sjl\(\)](#), [so\(\)](#), [tbt\(\)](#)

Examples

```
## Scalar example

sd <- lubridate::dhours(6)
napd <- lubridate::dhours(0.5)
nap <- TRUE
sd24(sd, napd, nap)
#> [1] "23400s (~6.5 hours)" # Expected

sd <- lubridate::dhours(9)
napd <- lubridate::dhours(1.5)
nap <- TRUE
sd24(sd, napd, nap)
#> [1] "37800s (~10.5 hours)" # Expected

sd <- lubridate::dhours(6.5)
napd <- lubridate::as.duration(NA)
nap <- FALSE
sd24(sd, napd, nap)
#> [1] "23400s (~6.5 hours)" # Expected

sd <- lubridate::as.duration(NA)
napd <- lubridate::dhours(2.3)
nap <- TRUE
sd24(sd, napd, nap)
#> [1] NA # Expected

## Vector example
```



```
sd <- c(lubridate::dhours(7.5), lubridate::dhours(8))
napd <- c(lubridate::dhours(0.75), lubridate::dhours(1))
nap <- c(TRUE, TRUE)
sd24(sd, napd, nap)
#> [1] "29700s (~8.25 hours)" "32400s (~9 hours)" # Expected
```

sdu

Compute MCTQ sleep duration

Description

[Maturing]

sdu() computes the **sleep duration** for standard, micro, and shift versions of the Munich Chrono-Type Questionnaire (MCTQ).

Please note that, although we tried to preserve the original authors' naming pattern for the MCTQ functions, the name `sd` provokes a dangerous name collision with the widely used `sd()` function (standard deviation). That's why we named it `sdu`. `sdu()` and `mstl()` are the only exceptions, all the other `mctq` functions maintain a strong naming resemblance with the original authors' naming pattern.

Usage

```
sdu(so, se)
```

Arguments

- | | |
|----|--|
| so | An hms object corresponding to the local time of sleep onset from a standard, micro, or shift version of the MCTQ questionnaire. You can use <code>so()</code> to compute it for the standard or shift version. |
| se | An hms object corresponding to the local time of sleep end from a standard, micro, or shift version of the MCTQ questionnaire. |

Details

Standard MCTQ functions were created following the guidelines in Roenneberg, Wirz-Justice, & Mellow (2003), Roenneberg, Allebrandt, Mellow, & Vetter (2012), and from The Worldwide Experimental Platform (theWeP, n.d.).

μ **MCTQ** functions were created following the guidelines in Ghotbi et al. (2020), in addition to the guidelines used for the standard MCTQ.

MCTQ^{Shift} functions were created following the guidelines in Juda, Vetter, & Roenneberg (2013), in addition to the guidelines used for the standard MCTQ.

See the References section to learn more.

Class requirements:

The `mctq` package works with a set of object classes specially created to hold time values. These classes can be found in the `lubridate` and `hms` packages. Please refer to those package documentations to learn more about them.

Rounding and fractional time:

Some operations may produce an output with fractional time (e.g., "19538.3828571429s (~5.43 hours)", 01:15:44.505). If you want, you can round it with `round_time()`.

Our recommendation is to avoid rounding, but, if you do, make sure that you only round your values after all computations are done. That way you avoid **round-off errors**.

Value

A `Duration` object corresponding to the vectorized difference between `se` and `so` in a circular time frame of 24 hours.

Guidelines

Roenneberg, Allebrandt, Merrow, & Vetter (2012), Ghotbi et al. (2020), Juda, Vetter, & Roenneberg (2013), and The Worldwide Experimental Platform (n.d.) guidelines for `sdu()` (*SD*) computation are as follows.

Notes:

- This computation must be applied to each section of the questionnaire.
- If you are visualizing this documentation in plain text, you may have some trouble understanding the equations. You can see this documentation on the package [website](#).

For standard and micro versions of the MCTQ:

$$SE_{W/F} - SO_{W/F}$$

Where:

- $SE_{W/F}$ = Local time of sleep end on work **or** work-free days.
- $SO_{W/F}$ = Local time of sleep onset on work **or** work-free days.

* W = Workdays; F = Work-free days.

For the shift version of the MCTQ:

$$SE_{W/F}^{M/E/N} - SO_{W/F}^{M/E/N}$$

Where:

- $SE_{W/F}^{M/E/N}$ = Local time of sleep end between two days in a particular shift **or** between two free days after a particular shift.
- $SO_{W/F}^{M/E/N}$ = Local time of sleep onset between two days in a particular shift **or** between two free days after a particular shift.

* W = Workdays; F = Work-free days, M = Morning shift; E = Evening shift; N = Night shift.

References

- Ghotbi, N., Pilz, L. K., Winnebeck, E. C., Vetter, C., Zerbini, G., Lenssen, D., Frighetto, G., Salamanca, M., Costa, R., Montagnese, S., & Roenneberg, T. (2020). The μ MCTQ: an ultra-short version of the Munich ChronoType Questionnaire. *Journal of Biological Rhythms*, 35(1), 98-110. doi: [10.1177/0748730419886986](https://doi.org/10.1177/0748730419886986).
- Juda, M., Vetter, C., & Roenneberg, T. (2013). The Munich ChronoType Questionnaire for shift-workers (MCTQ^{Shift}). *Journal of Biological Rhythms*, 28(2), 130-140. doi: [10.1177/0748730412475041](https://doi.org/10.1177/0748730412475041).
- Roenneberg T., Allebrandt K. V., Merrow M., & Vetter C. (2012). Social jetlag and obesity. *Current Biology*, 22(10), 939-43. doi: [10.1016/j.cub.2012.03.038](https://doi.org/10.1016/j.cub.2012.03.038).
- Roenneberg, T., Wirz-Justice, A., & Merrow, M. (2003). Life between clocks: daily temporal patterns of human chronotypes. *Journal of Biological Rhythms*, 18(1), 80-90. doi: [10.1177/0748730402239679](https://doi.org/10.1177/0748730402239679).
- The Worldwide Experimental Platform (n.d.). MCTQ. <https://www.thewep.org/documentations/mctq/>

See Also

Other MCTQ functions: [fd\(\)](#), [gu\(\)](#), [le_week\(\)](#), [msf_sc\(\)](#), [msl\(\)](#), [napd\(\)](#), [sd24\(\)](#), [sd_overall\(\)](#), [sd_week\(\)](#), [sjl_sc\(\)](#), [sjl_weighted\(\)](#), [sjl\(\)](#), [so\(\)](#), [tbt\(\)](#)

Examples

```
## Scalar example

so <- hms::parse_hm("23:00")
se <- hms::parse_hm("08:00")
sdu(so, se)
#> [1] "32400s (~9 hours)" # Expected

so <- hms::parse_hm("02:00")
se <- hms::parse_hm("12:30")
sdu(so, se)
#> [1] "37800s (~10.5 hours)" # Expected

so <- hms::parse_hm("03:15")
se <- hms::as_hms(NA)
sdu(so, se)
#> [1] NA # Expected

## Vector example

so <- c(hms::parse_hm("04:12"), hms::parse_hm("21:20"))
se <- c(hms::parse_hm("14:30"), hms::parse_hm("03:45"))
sdu(so, se)
#> [1] "37080s (~10.3 hours)" "23100s (~6.42 hours)" # Expected
```

sd_overall *Compute MCTQ overall sleep duration (only for MCTQ^{Shift})*

Description

[Maturing]

sd_overall() computes the **overall sleep duration in a particular shift** for the shift version of the Munich ChronoType Questionnaire (MCTQ).

See [sd_week\(\)](#) to compute the average weekly sleep duration for the standard and micro versions of the MCTQ.

Usage

```
sd_overall(sd_w, sd_f, n_w, n_f)
```

Arguments

sd_w	A Duration object corresponding to the sleep duration between two days in a particular shift from a shift version of the MCTQ questionnaire. You can use sdu() to compute it.
sd_f	A Duration object corresponding to the sleep duration between two free days after a particular shift from a shift version of the MCTQ questionnaire. You can use sdu() to compute it.
n_w	An integerish numeric object or an integer object corresponding to the number of days worked in a particular shift within a shift cycle from a shift version of the MCTQ questionnaire.
n_f	An integerish numeric object or an integer object corresponding to the number of free days after a particular shift within a shift cycle from a shift version of the MCTQ questionnaire.

Details

Standard MCTQ functions were created following the guidelines in Roenneberg, Wirz-Justice, & Mewes (2003), Roenneberg, Allebrandt, Mewes, & Vetter (2012), and from The Worldwide Experimental Platform (theWeP, n.d.).

μ MCTQ functions were created following the guidelines in Ghotbi et al. (2020), in addition to the guidelines used for the standard MCTQ.

MCTQ^{Shift} functions were created following the guidelines in Juda, Vetter, & Roenneberg (2013), in addition to the guidelines used for the standard MCTQ.

See the References section to learn more.

Class requirements:

The mctq package works with a set of object classes specially created to hold time values. These classes can be found in the [lubridate](#) and [hms](#) packages. Please refer to those package documents to learn more about them.

Rounding and fractional time:

Some operations may produce an output with fractional time (e.g., "19538.3828571429s (~5.43 hours)", 01:15:44.505). If you want, you can round it with `round_time()`.

Our recommendation is to avoid rounding, but, if you do, make sure that you only round your values after all computations are done. That way you avoid **round-off errors**.

Value

A `Duration` object corresponding to the vectorized weighted mean of `sd_w` and `sd_f` with `n_w` and `n_f` as weights.

Guidelines

Juda, Vetter, & Roenneberg (2013) and The Worldwide Experimental Platform (n.d.) guidelines for `sd_overall()` ($\emptyset SD^{M/E/N}$) computation are as follows.

Notes:

- This computation must be applied to each section of the questionnaire. If you're using the three-shift design proposed by the MCTQ authors, you need to compute three overall sleep duration (e.g., $\emptyset SD^M$; $\emptyset SD^E$; $\emptyset SD^N$).
- The overall sleep duration is the weighted average of the shift-specific mean sleep durations.
- If you are visualizing this documentation in plain text, you may have some trouble understanding the equations. You can see this documentation on the package [website](#).

Computation:

$$\frac{SD_W^{M/E/N} \times n_W^{M/E/N} + SD_F^{M/E/N} \times n_F^{M/E/N}}{n_W^{M/E/N} + n_F^{M/E/N}}$$

Where:

- $SD_W^{M/E/N}$ = Sleep duration between two days in a particular shift.
- $SD_F^{M/E/N}$ = Sleep duration between two free days after a particular shift.
- $n_W^{M/E/N}$ = Number of days worked in a particular shift within a shift cycle.
- $n_F^{M/E/N}$ = Number of free days after a particular shift within a shift cycle.

* W = Workdays; F = Work-free days, M = Morning shift; E = Evening shift; N = Night shift.

References

- Ghotbi, N., Pilz, L. K., Winnebeck, E. C., Vetter, C., Zerbini, G., Lensen, D., Frighetto, G., Salamanca, M., Costa, R., Montagnese, S., & Roenneberg, T. (2020). The μ MCTQ: an ultra-short version of the Munich ChronoType Questionnaire. *Journal of Biological Rhythms*, 35(1), 98-110. doi: [10.1177/0748730419886986](https://doi.org/10.1177/0748730419886986).
- Juda, M., Vetter, C., & Roenneberg, T. (2013). The Munich ChronoType Questionnaire for shift-workers (MCTQ^{Shift}). *Journal of Biological Rhythms*, 28(2), 130-140. doi: [10.1177/0748730412475041](https://doi.org/10.1177/0748730412475041).
- Roenneberg T., Allebrandt K. V., Mellow M., & Vetter C. (2012). Social jetlag and obesity. *Current Biology*, 22(10), 939-43. doi: [10.1016/j.cub.2012.03.038](https://doi.org/10.1016/j.cub.2012.03.038).

Roenneberg, T., Wirz-Justice, A., & Merrow, M. (2003). Life between clocks: daily temporal patterns of human chronotypes. *Journal of Biological Rhythms*, 18(1), 80-90. doi: [10.1177/0748730402239679](https://doi.org/10.1177/0748730402239679).

The Worldwide Experimental Platform (n.d.). MCTQ. <https://www.thewep.org/documentations/mctq/>

See Also

Other MCTQ functions: [fd\(\)](#), [gu\(\)](#), [le_week\(\)](#), [msf_sc\(\)](#), [msl\(\)](#), [napd\(\)](#), [sd24\(\)](#), [sd_week\(\)](#), [sdu\(\)](#), [sjl_sc\(\)](#), [sjl_weighted\(\)](#), [sjl\(\)](#), [so\(\)](#), [tbt\(\)](#)

Examples

```
## Scalar example

sd_w <- lubridate::dhours(5)
sd_f <- lubridate::dhours(9)
n_w <- 2
n_f <- 2
sd_overall(sd_w, sd_f, n_w, n_f)
#> [1] "25200s (~7 hours)" # Expected

sd_w <- lubridate::dhours(3.45)
sd_f <- lubridate::dhours(10)
n_w <- 3
n_f <- 1
sd_overall(sd_w, sd_f, n_w, n_f)
#> [1] "18315s (~5.09 hours)" # Expected

sd_w <- lubridate::as.duration(NA)
sd_f <- lubridate::dhours(12)
n_w <- 4
n_f <- 4
sd_overall(sd_w, sd_f, n_w, n_f)
#> [1] NA # Expected

## Vector example

sd_w <- c(lubridate::dhours(4), lubridate::dhours(7))
sd_f <- c(lubridate::dhours(12), lubridate::dhours(9))
n_w <- c(3, 4)
n_f <- c(2, 4)
sd_overall(sd_w, sd_f, n_w, n_f)
#> [1] "25920s (~7.2 hours)" "28800s (~8 hours)" # Expected

## Checking second output from vector example

if (requireNamespace("stats", quietly = TRUE)) {
  i <- 2
  x <- c(sd_w[i], sd_f[i])
  w <- c(n_w[i], n_f[i])
  lubridate::as.duration(stats::weighted.mean(x, w))
}
```

```

}
#> [1] "28800s (~8 hours)" # Expected

## Converting the output to 'hms'

sd_w <- lubridate::dhours(4.75)
sd_f <- lubridate::dhours(10)
n_w <- 5
n_f <- 2
sd_overall(sd_w, sd_f, n_w, n_f)
#> [1] "22500s (~6.25 hours)" # Expected

hms::as_hms(as.numeric(sd_overall(sd_w, sd_f, n_w, n_f)))
#> 06:15:00 # Expected

## Rounding the output at the seconds level

sd_w <- lubridate::dhours(5.9874)
sd_f <- lubridate::dhours(9.3)
n_w <- 3
n_f <- 2
sd_overall(sd_w, sd_f, n_w, n_f)
#> [1] "26324.784s (~7.31 hours)" # Expected

round_time(sd_overall(sd_w, sd_f, n_w, n_f))
#> [1] "26325s (~7.31 hours)" # Expected

```

sd_week

Compute MCTQ average weekly sleep duration

Description

[Maturing]

sd_week() computes the **average weekly sleep duration** for the standard and micro versions of the Munich ChronoType Questionnaire (MCTQ).

See [sd_overall\(\)](#) to compute the overall sleep duration of a particular shift for the shift version of the MCTQ.

Usage

```
sd_week(sd_w, sd_f, wd)
```

Arguments

sd_w A [Duration](#) object corresponding to the **sleep duration on workdays** from a standard or micro version of the MCTQ questionnaire. You can use [sdu\(\)](#) to compute it.

sd_f	A Duration object corresponding to the sleep duration on work-free days from a standard or micro version of the MCTQ questionnaire. You can use sdu() to compute it.
wd	An integerish numeric object or an integer object corresponding to the number of workdays per week from a standard or micro version of the MCTQ questionnaire.

Details

Standard MCTQ functions were created following the guidelines in Roenneberg, Wirz-Justice, & Mellow (2003), Roenneberg, Allebrandt, Mellow, & Vetter (2012), and from The Worldwide Experimental Platform (theWeP, n.d.).

μ **MCTQ** functions were created following the guidelines in Ghotbi et al. (2020), in addition to the guidelines used for the standard MCTQ.

MCTQ^{Shift} functions were created following the guidelines in Juda, Vetter, & Roenneberg (2013), in addition to the guidelines used for the standard MCTQ.

See the References section to learn more.

Class requirements:

The `mctq` package works with a set of object classes specially created to hold time values. These classes can be found in the [lubridate](#) and [hms](#) packages. Please refer to those package documents to learn more about them.

Rounding and fractional time:

Some operations may produce an output with fractional time (e.g., "19538.3828571429s (~5.43 hours)", 01:15:44.505). If you want, you can round it with [round_time\(\)](#).

Our recommendation is to avoid rounding, but, if you do, make sure that you only round your values after all computations are done. That way you avoid **round-off errors**.

Value

A [Duration](#) object corresponding to the vectorized weighted mean of `sd_w` and `sd_f` with `wd` and `fd(wd)` as weights.

Guidelines

Roenneberg, Allebrandt, Mellow, & Vetter (2012), Ghotbi et al. (2020), and The Worldwide Experimental Platform (n.d.) guidelines for `sd_week()` (SD_{week}) computation are as follows.

Notes:

- The average weekly sleep duration is the weighted average of the sleep durations on work and work-free days in a week.
- If you are visualizing this documentation in plain text, you may have some trouble understanding the equations. You can see this documentation on the package [website](#).

Computation:

$$\frac{SD_W \times WD + SD_F \times FD}{7}$$

Where:

- SD_W = Sleep duration on workdays.
- SD_F = Sleep duration on work-free days.
- WD = Number of workdays per week ("I have a regular work schedule and work ___ days per week").
- FD = Number of work-free days per week.

* W = Workdays; F = Work-free days.

References

- Ghotbi, N., Pilz, L. K., Winnebeck, E. C., Vetter, C., Zerbini, G., Lenssen, D., Frighetto, G., Salamanca, M., Costa, R., Montagnese, S., & Roenneberg, T. (2020). The μ MCTQ: an ultra-short version of the Munich ChronoType Questionnaire. *Journal of Biological Rhythms*, 35(1), 98-110. doi: [10.1177/0748730419886986](https://doi.org/10.1177/0748730419886986).
- Juda, M., Vetter, C., & Roenneberg, T. (2013). The Munich ChronoType Questionnaire for shift-workers (MCTQ^{Shift}). *Journal of Biological Rhythms*, 28(2), 130-140. doi: [10.1177/0748730412475041](https://doi.org/10.1177/0748730412475041).
- Roenneberg T., Allebrandt K. V., Meroow M., & Vetter C. (2012). Social jetlag and obesity. *Current Biology*, 22(10), 939-43. doi: [10.1016/j.cub.2012.03.038](https://doi.org/10.1016/j.cub.2012.03.038).
- Roenneberg, T., Wirz-Justice, A., & Meroow, M. (2003). Life between clocks: daily temporal patterns of human chronotypes. *Journal of Biological Rhythms*, 18(1), 80-90. doi: [10.1177/0748730402239679](https://doi.org/10.1177/0748730402239679).
- The Worldwide Experimental Platform (n.d.). MCTQ. <https://www.thewep.org/documentations/mctq/>

See Also

Other MCTQ functions: [fd\(\)](#), [gu\(\)](#), [le_week\(\)](#), [msf_sc\(\)](#), [msl\(\)](#), [napd\(\)](#), [sd24\(\)](#), [sd_overall\(\)](#), [sdu\(\)](#), [sjl_sc\(\)](#), [sjl_weighted\(\)](#), [sjl\(\)](#), [so\(\)](#), [tbt\(\)](#)

Examples

```
## Scalar example

sd_w <- lubridate::dhours(4)
sd_f <- lubridate::dhours(8)
wd <- 5
sd_week(sd_w, sd_f, wd)
#> [1] "18514.2857142857s (~5.14 hours)" # Expected

sd_w <- lubridate::dhours(7)
sd_f <- lubridate::dhours(7)
wd <- 4
sd_week(sd_w, sd_f, wd)
#> [1] "25200s (~7 hours)" # Expected

sd_w <- lubridate::as.duration(NA)
sd_f <- lubridate::dhours(10)
```

```

wd <- 6
sd_week(sd_w, sd_f, wd)
#> [1] NA # Expected

## Vector example

sd_w <- c(lubridate::dhours(4.5), lubridate::dhours(5.45))
sd_f <- c(lubridate::dhours(8), lubridate::dhours(7.3))
wd <- c(3, 7)
sd_week(sd_w, sd_f, wd)
#> [1] "23400s (~6.5 hours)" "19620s (~5.45 hours)" # Expected

## Checking second output from vector example

if (requireNamespace("stats", quietly = TRUE)) {
  i <- 2
  x <- c(sd_w[i], sd_f[i])
  w <- c(wd[i], fd(wd[i]))
  lubridate::as.duration(stats::weighted.mean(x, w))
}
#> [1] "19620s (~5.45 hours)" # Expected

## Converting the output to 'hms'

sd_w <- lubridate::dhours(5.45)
sd_f <- lubridate::dhours(9.5)
wd <- 5
x <- sd_week(sd_w, sd_f, wd)
x
#> [1] "23785.7142857143s (~6.61 hours)" # Expected
hms::as_hms(as.numeric(x))
#> 06:36:25.714286 # Expected

## Rounding the output at the seconds level

sd_w <- lubridate::dhours(4.5)
sd_f <- lubridate::dhours(7.8)
wd <- 3
sd_week(sd_w, sd_f, wd)
#> [1] "22988.5714285714s (~6.39 hours)" # Expected

round_time(sd_week(sd_w, sd_f, wd))
#> [1] "22989s (~6.39 hours)" # Expected

```

shift_mctq

A fictional MCTQ[^] Shift dataset

Description

[Maturing]

A fictional dataset, for **testing and learning purposes**, composed of basic/measurable and computed variables of the Munich ChronoType Questionnaire (MCTQ) shift version.

This data was created following the guidelines in Juda, Vetter, & Roenneberg (2013), in addition to the guidelines found in Roenneberg, Wirz-Justice, & Merrow (2003), Roenneberg, Allebrandt, Merrow, & Vetter (2012), Jankowski (2017), and The Worldwide Experimental Platform (n.d.). See the References and Details sections to learn more.

Usage

```
shift_mctq
```

Format

A [tibble](#) with 135 columns and 50 rows:

id A unique [integer](#) value to identify each respondent in the dataset.

Type: Control.

R class: [integer](#).

n_w_m Number of days **worked in morning shifts** within a shift cycle.

Type: Basic.

R class: [integer](#).

bt_w_m Local time of going to bed on workdays **between two morning shifts**.

Statement (EN): "I go to bed at ___ o'clock".

Type: Basic.

R class: [hms](#).

sprep_w_m Local time of preparing to sleep on workdays **between two morning shifts**.

Statement (EN): "I actually get ready to fall asleep at ___ o'clock".

Type: Basic.

R class: [hms](#).

slat_w_m Sleep latency or time to fall asleep after preparing to sleep on workdays **between two morning shifts**.

Statement (EN): "I need ___ minutes to fall asleep".

Type: Basic.

R class: [Duration](#).

so_w_m Local time of sleep onset on workdays **between two morning shifts**.

Type: Computed.

R class: *hms*.

se_w_m Local time of sleep end on workdays **between two morning shifts**.

Statement (EN): "I wake up at ___ o'clock".

Type: Basic.

R class: *hms*.

tgu_w_m Time to get up on workdays **between two morning shifts**.

Statement (EN): "I get up after ___ minutes".

Type: Basic.

R class: *Duration*.

gu_w_m Local time of getting out of bed on workdays **between two morning shifts**.

Type: Computed.

R class: *hms*.

alarm_w_m A *logical* value indicating if the respondent uses an alarm clock to wake up on workdays **between two morning shifts**.

Statement (EN): "I wake up at ___ o'clock: (___) with alarm (___) without alarm".

Type: Basic.

R class: *logical*.

reasons_w_m A *logical* value indicating if the respondent has any particular reasons for why they **cannot** freely choose their sleep times on workdays **between two morning shifts**.

Statement (EN): "There are particular reasons why I **cannot** freely choose my sleep times on morning shifts: Yes (___) No (___)".

Type: Basic.

R class: *logical*.

reasons_why_w_m Particular reasons for why the respondent cannot freely choose their sleep times on workdays **between two morning shifts**.

Statement (EN): "If "Yes": Child(ren)/pet(s) (___) Hobbies (___) Others, for example: ___".

Type: Basic.

R class: [character](#).

sd_w_m Sleep duration on workdays **between two morning shifts**.

Type: Computed.

R class: [Duration](#).

tbt_w_m Total time in bed on workdays **between two morning shifts**.

Type: Computed.

R class: [Duration](#).

msw_m Local time of mid-sleep on workdays **between two morning shifts**.

Type: Computed.

R class: [hms](#).

nap_w_m A [logical](#) value indicating if the respondent usually takes a nap on workdays **between two morning shifts**.

Statement (EN): "I usually take a nap: Yes (___) No (___)".

Type: Basic.

R class: [logical](#).

napo_w_m Local time of nap onset on workdays **between two morning shifts**.

Statement (EN): "If "Yes": I take a nap from ___ o'clock to ___ o'clock".

Type: Basic.

R class: [hms](#).

nape_w_m Local time of nap end on workdays **between two morning shifts**.

Statement (EN): "If "Yes": I take a nap from ___ o'clock to ___ o'clock".

Type: Basic.

R class: [hms](#).

napd_w_m Nap duration on workdays **between two morning shifts**.

Type: Computed.

R class: [Duration](#).

sd24_w_m 24 hours sleep duration (sleep duration + nap duration) on workdays **between two morning shifts**.

Type: Computed.

R class: [Duration](#).

n_fm Number of free days **after working in morning shifts** within a shift cycle.

Type: Basic.

R class: [integer](#).

bt_fm Local time of going to bed on work-free days **between two free days after morning shifts**.

Statement (EN): "I go to bed at ___ o'clock".

Type: Basic.

R class: [hms](#).

sprep_fm Local time of preparing to sleep on work-free days **between two free days after morning shifts**.

Statement (EN): "I actually get ready to fall asleep at ___ o'clock".

Type: Basic.

R class: [hms](#).

slat_fm Sleep latency or time to fall asleep after preparing to sleep on work-free days **between two free days after morning shifts**.

Statement (EN): "I need ___ minutes to fall asleep".

Type: Basic.

R class: [Duration](#).

so_fm Local time of sleep onset on work-free days **between two free days after morning shifts**.

Type: Computed.

R class: [hms](#).

se_fm Local time of sleep end on work-free days **between two free days after morning shifts**.

Statement (EN): "I wake up at ___ o'clock".

Type: Basic.

R class: [hms](#).

tgu_fm Time to get up on work-free days **between two free days after morning shifts**.

Statement (EN): "I get up after ___ minutes".

Type: Basic.

R class: [Duration](#).

gu_f_m Local time of getting out of bed on work-free days **between two free days after morning shifts**.

Type: Computed.

R class: [hms](#).

alarm_f_m A [logical](#) value indicating if the respondent uses an alarm clock to wake up on work-free days **between two free days after morning shifts**.

Statement (EN): "I wake up at ___ o'clock: (___) with alarm (___) without alarm".

Type: Basic.

R class: [logical](#).

reasons_f_m A [logical](#) value indicating if the respondent has any particular reasons for why they **cannot** freely choose their sleep times on work-free days **between two free days after morning shifts**.

Statement (EN): "There are particular reasons why I **cannot** freely choose my sleep times on morning shifts: Yes (___) No (___)".

Type: Basic.

R class: [logical](#).

reasons_why_f_m Particular reasons for why the respondent cannot freely choose their sleep times on work-free days **between two free days after morning shifts**.

Statement (EN): "If "Yes": Child(ren)/pet(s) (___) Hobbies (___) Others, for example: ___".

Type: Basic.

R class: [character](#).

sd_f_m Sleep duration on work-free days **between two free days after morning shifts**.

Type: Computed.

R class: [Duration](#).

tbt_f_m Total time in bed on work-free days **between two free days after morning shifts**.

Type: Computed.

R class: [Duration](#).

msf_m Local time of mid-sleep on work-free days **between two free days after morning shifts.**

Type: Computed.

R class: `hms`.

nap_f_m A `logical` value indicating if the respondent usually takes a nap on work-free days **between two free days after morning shifts.**

Statement (EN): "I usually take a nap: Yes (___) No (___)".

Type: Basic.

R class: `logical`.

napo_f_m Local time of nap onset on work-free days **between two free days after morning shifts.**

Statement (EN): "If "Yes": I take a nap from ___ o'clock to ___ o'clock".

Type: Basic.

R class: `hms`.

nape_f_m Local time of nap end on work-free days **between two free days after morning shifts.**

Statement (EN): "If "Yes": I take a nap from ___ o'clock to ___ o'clock".

Type: Basic.

R class: `hms`.

napd_f_m Nap duration on work-free days **between two free days after morning shifts.**

Type: Computed.

R class: `Duration`.

sd24_f_m 24 hours sleep duration (sleep duration + nap duration) on work-free days **between two free days after morning shifts.**

Type: Computed.

R class: `Duration`.

sd_overall_m Overall sleep duration considering workdays **between two morning shifts** and work-free days **between two free days after morning shifts.**

Type: Computed.

R class: `Duration`.

msf_sc_m Corrected local time of mid-sleep on work-free days **between two free days after morning shifts.**

Type: Computed.

R class: `hms`.

sjl_rel_m Relative social jetlag considering workdays **between two morning shifts** and work-free days **between two free days after morning shifts**.

Type: Computed.

R class: `Duration`.

sjl_m Absolute social jetlag considering workdays **between two morning shifts** and work-free days **between two free days after morning shifts**.

Type: Computed.

R class: `Duration`.

sjl_sc_rel_m Jankowski's relative sleep-corrected social jetlag considering workdays **between two morning shifts** and work-free days **between two free days after morning shifts**.

Type: Computed.

R class: `Duration`.

sjl_sc_m Jankowski's sleep-corrected social jetlag considering workdays **between two morning shifts** and work-free days **between two free days after morning shifts**.

Type: Computed.

R class: `Duration`.

... For brevity, the subsequent variables, except for **sjl_weighted** and **sjl_sc_weighted** (described below), are not shown here. That's because they have the same configurations of the variables shown above, differing only by shift (**evening shift** (`_e`) and **night shift** (`_n`)).

sjl_weighted Absolute social jetlag across all shifts.

Type: Computed.

R class: `Duration`.

#'

sjl_sc_weighted Jankowski's sleep-corrected social jetlag across all shifts.

Type: Computed.

R class: `Duration`.

Details

shift_mctq is a tidied, validated, and transformed version of `raw_data("shift_mctq.csv")`.

Guidelines:

To learn more about the Munich ChronoType Questionnaire (MCTQ), see Roenneberg, Wirz-Justice, & Mellow (2003), Roenneberg, Allebrandt, Mellow, & Vetter (2012), Roenneberg et al. (2015), and Roenneberg, Pilz, Zerbini, & Winnebeck (2019).

To know about different MCTQ versions, see Juda, Vetter, & Roenneberg (2013) and Ghotbi et al. (2020).

To learn about the sleep-corrected social jetlag, see Jankowski (2017).

If you're curious about the variable computations and want to have access to the full questionnaire, see The Worldwide Experimental Platform (n.d.).

Data building and data wrangling:

This dataset was created by randomized sampling (see `random_mctq()`) and by manual insertions of special cases. Its purpose is to demonstrate common cases and data issues that researchers may find in their MCTQ data, in addition to be a suggested data structure for MCTQ data.

You can see the `shift_mctq` build and data wrangling processes [here](#).

Variable naming:

The naming of the variables took into account the naming scheme used in MCTQ publications, in addition to the guidelines of the [tidyverse style guide](#).

Variable classes:

The `mctq` package works with a set of object classes specially created to hold time values. These classes can be found in the [hms](#) and [lubridate](#) package.

Duration objects:

If you prefer to view `Duration` objects as `hms` objects, run `pretty_mctq(shift_mctq)`.

Source

Created by Daniel Vartanian (package author).

References

- Ghotbi, N., Pilz, L. K., Winnebeck, E. C., Vetter, C., Zerbini, G., Lenssen, D., Frighetto, G., Salamanca, M., Costa, R., Montagnese, S., & Roenneberg, T. (2020). The μ MCTQ: an ultra-short version of the Munich ChronoType Questionnaire. *Journal of Biological Rhythms*, 35(1), 98-110. doi: [10.1177/0748730419886986](https://doi.org/10.1177/0748730419886986).
- Jankowski K. S. (2017). Social jet lag: sleep-corrected formula. *Chronobiology International*, 34(4), 531-535. doi: [10.1080/07420528.2017.1299162](https://doi.org/10.1080/07420528.2017.1299162).
- Juda, M., Vetter, C., & Roenneberg, T. (2013). The Munich ChronoType Questionnaire for shift-workers (MCTQ^{Shift}). *Journal of Biological Rhythms*, 28(2), 130-140. doi: [10.1177/0748730412475041](https://doi.org/10.1177/0748730412475041).
- Roenneberg T., Allebrandt K. V., Mellow M., & Vetter C. (2012). Social jetlag and obesity. *Current Biology*, 22(10), 939-43. doi: [10.1016/j.cub.2012.03.038](https://doi.org/10.1016/j.cub.2012.03.038).
- Roenneberg, T., Keller, L. K., Fischer, D., Matera, J. L., Vetter, C., & Winnebeck, E. C. (2015). Human activity and rest in situ. In A. Sehgal (Ed.), *Methods in Enzymology* (Vol. 552, pp. 257-283). London, UK: Academic Press. doi: [10.1016/bs.mie.2014.11.028](https://doi.org/10.1016/bs.mie.2014.11.028).
- Roenneberg, T., Pilz, L. K., Zerbini, G., & Winnebeck, E. C. (2019). Chronotype and social jetlag: a (self-) critical review. *Biology*, 8(3), 54. doi: [10.3390/biology8030054](https://doi.org/10.3390/biology8030054).

Roenneberg, T., Wirz-Justice, A., & Merrow, M. (2003). Life between clocks: daily temporal patterns of human chronotypes. *Journal of Biological Rhythms*, 18(1), 80-90. doi: [10.1177/0748730402239679](https://doi.org/10.1177/0748730402239679).

The Worldwide Experimental Platform (n.d.). MCTQ. <https://www.thewep.org/documentations/mctq/>

See Also

Other datasets: [micro_mctq](#), [std_mctq](#)

shorter_interval *Find the shorter or longer interval between two hours*

Description

[Maturing]

shorter_interval() returns the shorter interval between two [hms](#) or [POSIXt](#) object hours.

longer_interval() do the inverse of shorter_interval(), i.e., returns the longer interval between two hours.

shorter_duration() and longer_duration() return the interval time span of shorter_interval() and longer_interval() as [Duration](#) objects.

Usage

```
shorter_interval(x, y)
```

```
longer_interval(x, y)
```

```
shorter_duration(x, y)
```

```
longer_duration(x, y)
```

Arguments

x, y An [hms](#) or [POSIXt](#) object.

Details

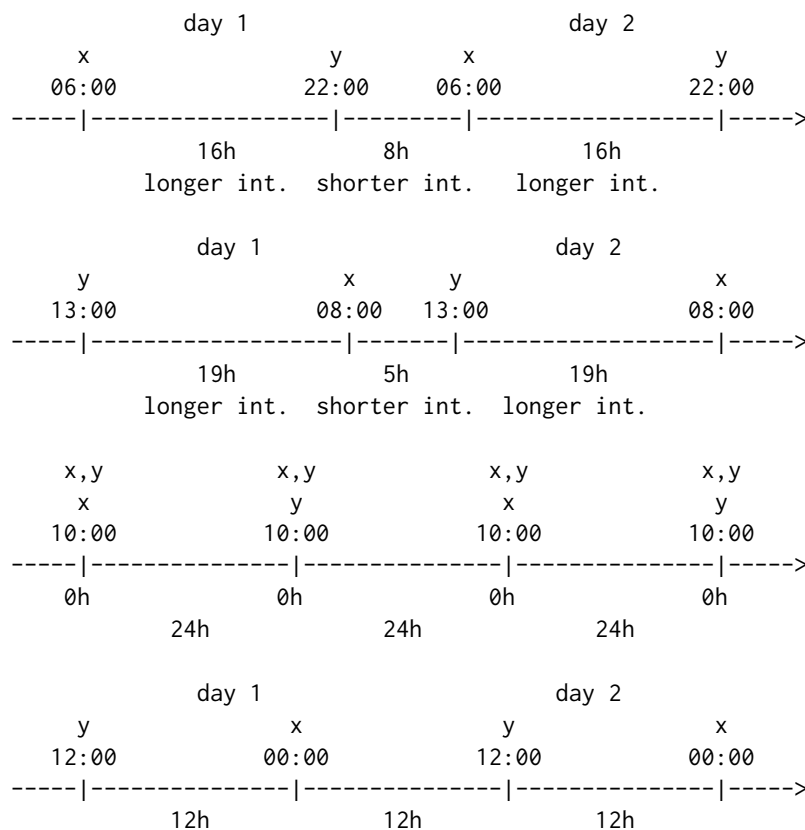
The two intervals problem:

Given two hours, x and y, in a two-day timeline, without date references, there will be always two possible intervals between them, as illustrated below.

To figure out what interval is the shorter or the longer, shorter_interval() and longer_interval() verify two scenarios: 1. When x comes before y; and 2. when x comes after y. This only works if x value is smaller than y, therefore, the function will make sure to swap x and y values if the latter assumption is not true.

Because `shorter_interval()` objective is to find the shorter interval, if `x` and `y` are equal, the shorter interval will have a length of 0 hours, resulting in an interval from `x` to `x`. But, if `longer_interval()` is used instead, the latter condition will return a interval with 24 hours of length (from `x` to `x + 1 day`).

In cases when `x` and `y` distance themselves by 12 hours, there will be no shorter or longer interval (they will have equal length). In these cases, `shorter_interval()` and `longer_interval()` will return the same value (an interval of 12 hours).



Class requirements:

The `mctq` package works with a set of object classes specially created to hold time values. These classes can be found in the [hms](#) and [lubridate](#) package.

Base date and timezone:

`shorter_interval()` and `longer_interval()` use the **Unix epoch** (1970-01-01) date as the start date for creating intervals.

The output will always have "UTC" set as timezone. Learn more about time zones in [?timezone](#).

POSIXt objects:

`POSIXt` objects passed as argument to `x` or `y` will be stripped of their dates. Only the time will be considered.

Both `POSIXct` and `POSIXlt` are objects that inherits the class `POSIXt`. Learn more about it in [?DateTimeClasses](#).

NA values:

shorter_interval() or longer_interval() will return an [Interval](#) NA-NA if x or y are NA.
 shorter_duration() or longer_duration() will return a [Duration](#) NA if x or y are NA.

Value

- For shorter_interval() or longer_interval(), an [Interval](#) object with the shorter or longer interval between x and y.
- For shorter_duration() or longer_duration(), a [Duration](#) object with the shorter or longer duration between x and y.

See Also

Other utility functions: [assign_date\(\)](#), [cycle_time\(\)](#), [pretty_mctq\(\)](#), [qplot_walk\(\)](#), [random_mctq\(\)](#), [raw_data\(\)](#), [round_time\(\)](#), [sum_time\(\)](#)

Examples

```
## Scalar example

x <- hms::parse_hm("23:00")
y <- hms::parse_hm("01:00")

shorter_interval(x, y)
#> [1] 1970-01-01 23:00:00 UTC--1970-01-02 01:00:00 UTC # Expected
shorter_duration(x, y)
#> [1] "7200s (~2 hours)" # Expected
longer_interval(x, y)
#> [1] 1970-01-01 01:00:00 UTC--1970-01-01 23:00:00 UTC # Expected
longer_duration(x, y)
#> [1] "79200s (~22 hours)" # Expected

x <- lubridate::as_datetime("1985-01-15 12:00:00")
y <- lubridate::as_datetime("2020-09-10 12:00:00")

shorter_interval(x, y)
#> [1] 1970-01-01 12:00:00 UTC--1970-01-01 12:00:00 UTC # Expected
shorter_duration(x, y)
#> [1] "0s" # Expected
longer_interval(x, y)
#> [1] 1970-01-01 12:00:00 UTC--1970-01-02 12:00:00 UTC # Expected
longer_duration(x, y)
#> [1] "86400s (~1 days)" # Expected

## Vector example

x <- c(hms::parse_hm("15:30"), hms::parse_hm("21:30"))
y <- c(hms::parse_hm("19:30"), hms::parse_hm("04:00"))

shorter_interval(x, y)
#> [1] 1970-01-01 15:30:00 UTC--1970-01-01 19:30:00 UTC # Expected
```

```
#> [2] 1970-01-01 21:30:00 UTC--1970-01-02 04:00:00 UTC # Expected
shorter_duration(x, y)
#> [1] [1] "14400s (~4 hours)" "23400s (~6.5 hours)" # Expected
longer_interval(x, y)
#> [1] 1970-01-01 19:30:00 UTC--1970-01-02 15:30:00 UTC # Expected
#> [2] 1970-01-01 04:00:00 UTC--1970-01-01 21:30:00 UTC # Expected
longer_duration(x, y)
#> [1] "72000s (~20 hours)" "63000s (~17.5 hours)" # Expected
```

 sjl

 Compute MCTQ social jetlag

Description

[Maturing]

sjl() computes the **relative or absolute social jetlag** for standard, micro, and shift versions of the Munich ChronoType Questionnaire (MCTQ).

sjl_rel() is just a wrapper for sjl() with abs = FALSE.

Usage

```
sjl(msw, msf, abs = TRUE, method = "shorter")
```

```
sjl_rel(msw, msf, method = "shorter")
```

Arguments

msw	An hms object corresponding to the local time of mid-sleep on workdays from a standard, micro, or shift version of the MCTQ questionnaire. You can use msl() to compute it.
msf	An hms object corresponding to the local time of mid-sleep on work-free days from a standard, micro, or shift version of the MCTQ questionnaire. You can use msl() to compute it.
abs	(optional) a logical object indicating if the function must return an absolute value (default: TRUE).
method	(optional) a string indicating which method the function must use to compute the social jetlag. See the Methods section to learn more (default: "shorter").

Details

Standard MCTQ functions were created following the guidelines in Roenneberg, Wirz-Justice, & Mellow (2003), Roenneberg, Allebrandt, Mellow, & Vetter (2012), and from The Worldwide Experimental Platform (theWeP, n.d.).

μ MCTQ functions were created following the guidelines in Ghotbi et al. (2020), in addition to the guidelines used for the standard MCTQ.

MCTQ^{Shift} functions were created following the guidelines in Juda, Vetter, & Roenneberg (2013), in addition to the guidelines used for the standard MCTQ.

See the References section to learn more.

Class requirements:

The `mctq` package works with a set of object classes specially created to hold time values. These classes can be found in the `lubridate` and `hms` packages. Please refer to those package documentations to learn more about them.

Rounding and fractional time:

Some operations may produce an output with fractional time (e.g., "19538.3828571429s (~5.43 hours)", "01:15:44.505"). If you want, you can round it with `round_time()`.

Our recommendation is to avoid rounding, but, if you do, make sure that you only round your values after all computations are done. That way you avoid **round-off errors**.

Value

- If `abs = TRUE`, a `Duration` object corresponding to the absolute social jetlag.
- If `abs = FALSE`, a `Duration` object corresponding to the relative social jetlag.

The output may also vary depending on the method used.

Guidelines

Roenneberg, Allebrandt, Merrow, & Vetter (2012), Juda, Vetter, & Roenneberg (2013), and The Worldwide Experimental Platform (n.d.) guidelines for `sjl()` (*SJL_{rel}* and *SJL*) computation are as follows.

Notes:

- For **MCTQ^{Shift}**, the computation below must be applied to each shift section of the questionnaire.
- Due to time arithmetic issues, `sjl()` does a slightly different computation by default than those proposed by the authors mentioned above. See `vignette("sjl-computation", package = "mctq")` for more details.
- If you are visualizing this documentation in plain text, you may have some trouble understanding the equations. You can see this documentation on the package [website](#).

For standard and micro versions of the MCTQ:

Relative social jetlag (*SJL_{rel}*):

$$MSF - MSW$$

Absolute social jetlag (*SJL*):

$$|MSF - MSW|$$

Where:

- MSW = Local time of mid-sleep on workdays.
- MSF = Local time of mid-sleep on work-free days.

* W = Workdays; F = Work-free days.

For the shift version of the MCTQ:

Relative social jetlag (SJL_{rel}):

$$MSF^{M/E/N} - MSW^{M/E/N}$$

Absolute social jetlag (SJL):

$$|MSF^{M/E/N} - MSW^{M/E/N}|$$

Where:

- $MSW^{M/E/N}$ = Local time of mid-sleep between two days in a particular shift.
 - $MSF^{M/E/N}$ = Local time of mid-sleep between two free days after a particular shift.
- * W = Workdays; F = Work-free days, M = Morning shift; E = Evening shift; N = Night shift.

Methods for computing the social jetlag

There are different approaches to compute the social jetlag (SJL). By default, `sjl()` uses an approach that we call "the shorter interval approach" ("shorter").

The topics below provide a simple explanation of each method supported by `sjl()`. To get a detail understating of this methods, see `vignette("sjl-computation", package = "mctq")`.

- "difference"

By using `method = "difference"`, `sjl()` will do the exact computation proposed by the MCTQ authors, i.e., SJL will be computed as the linear difference between MSF and MSW (see the Guidelines section).

We do not recommend using this method, as it has many limitations.

- "shorter"

This is the default method for `sjl()`. It's based on the shorter interval between MSW and MSF , solving most of the issues relating to SJL computation.

- "longer"

The "longer" method uses the same logic of the "shorter" method, but, instead of using the shorter interval between MSW and MSF , it uses the longer interval between the two, considering a two-day window.

This method may help in special contexts, like when dealing with shift-workers that have a greater than 12 hours distance between their mid-sleep hours.

References

- Ghotbi, N., Pilz, L. K., Winnebeck, E. C., Vetter, C., Zerbini, G., Lenssen, D., Frighetto, G., Salamanca, M., Costa, R., Montagnese, S., & Roenneberg, T. (2020). The μ MCTQ: an ultra-short version of the Munich ChronoType Questionnaire. *Journal of Biological Rhythms*, 35(1), 98-110. doi: [10.1177/0748730419886986](https://doi.org/10.1177/0748730419886986).
- Jankowski K. S. (2017). Social jet lag: sleep-corrected formula. *Chronobiology International*, 34(4), 531-535. doi: [10.1080/07420528.2017.1299162](https://doi.org/10.1080/07420528.2017.1299162).
- Juda, M., Vetter, C., & Roenneberg, T. (2013). The Munich ChronoType Questionnaire for shift-workers (MCTQ^{Shift}). *Journal of Biological Rhythms*, 28(2), 130-140. doi: [10.1177/0748730412475041](https://doi.org/10.1177/0748730412475041).
- Roenneberg T., Allebrandt K. V., Merrow M., & Vetter C. (2012). Social jetlag and obesity. *Current Biology*, 22(10), 939-43. doi: [10.1016/j.cub.2012.03.038](https://doi.org/10.1016/j.cub.2012.03.038).
- Roenneberg, T., Pilz, L. K., Zerbini, G., & Winnebeck, E. C. (2019). Chronotype and social jetlag: a (self-) critical review. *Biology*, 8(3), 54. doi: [10.3390/biology8030054](https://doi.org/10.3390/biology8030054).
- Roenneberg, T., Wirz-Justice, A., & Merrow, M. (2003). Life between clocks: daily temporal patterns of human chronotypes. *Journal of Biological Rhythms*, 18(1), 80-90. doi: [10.1177/0748730402239679](https://doi.org/10.1177/0748730402239679).
- The Worldwide Experimental Platform (n.d.). MCTQ. <https://www.thewep.org/documentations/mctq/>

See Also

Other MCTQ functions: [fd\(\)](#), [gu\(\)](#), [le_week\(\)](#), [msf_sc\(\)](#), [msl\(\)](#), [napd\(\)](#), [sd24\(\)](#), [sd_overall\(\)](#), [sd_week\(\)](#), [sdu\(\)](#), [sjl_sc\(\)](#), [sjl_weighted\(\)](#), [so\(\)](#), [tbt\(\)](#)

Examples

```
## Scalar example

msw <- hms::parse_hm("03:30")
msf <- hms::parse_hm("05:00")

sjl(msw, msf)
#> [1] "5400s (~1.5 hours)" # Expected
sjl(msw, msf, abs = FALSE)
#> [1] "5400s (~1.5 hours)" # Expected
sjl_rel(msw, msf) # Wrapper function
#> [1] "5400s (~1.5 hours)" # Expected

msw <- hms::parse_hm("04:30")
msf <- hms::parse_hm("23:30")

sjl(msw, msf)
#> [1] "18000s (~5 hours)" # Expected
sjl(msw, msf, abs = FALSE)
#> [1] "18000s (~-5 hours)" # Expected
sjl_rel(msw, msf) # Wrapper function
#> [1] "18000s (~-5 hours)" # Expected
```

```

msw <- hms::as_hms(NA)
msf <- hms::parse_hm("05:15")

sjl(msw, msf)
#> [1] NA # Expected

## Vector example

msw <- c(hms::parse_hm("02:05"), hms::parse_hm("04:05"))
msf <- c(hms::parse_hm("23:05"), hms::parse_hm("04:05"))

sjl(msw, msf)
#> [1] "10800s (~3 hours)" "0s" # Expected
sjl(msw, msf, abs = FALSE)
#> [1] "-10800s (~-3 hours)" "0s" # Expected
sjl_rel(msw, msf) # Wrapper function
#> [1] "-10800s (~-3 hours)" "0s" # Expected

## Using different methods

msw <- hms::parse_hm("19:15")
msf <- hms::parse_hm("02:30")

sjl(msw, msf, abs = FALSE, method = "difference")
#> [1] "-60300s (~-16.75 hours)" # Expected
sjl(msw, msf, abs = FALSE, method = "shorter") # Default method
#> [1] "26100s (~7.25 hours)" # Expected
sjl(msw, msf, abs = FALSE, method = "longer")
#> [1] "-60300s (~-16.75 hours)" # Expected

msw <- hms::parse_hm("02:45")
msf <- hms::parse_hm("04:15")

sjl(msw, msf, abs = FALSE, method = "difference")
#> [1] "5400s (~1.5 hours)" # Expected
sjl(msw, msf, abs = FALSE, method = "shorter") # Default method
#> [1] "5400s (~1.5 hours)" # Expected
sjl(msw, msf, abs = FALSE, method = "longer")
#> [1] "-8100s (~-2.25 hours)" # Expected

## Converting the output to 'hms'

msw <- hms::parse_hm("01:15")
msf <- hms::parse_hm("03:25")
sjl(msw, msf)
#> [1] "7800s (~2.17 hours)" # Expected

hms::as_hms(as.numeric(sjl(msw, msf)))
#> 02:10:00 # Expected

## Rounding the output at the seconds level

msw <- hms::parse_hms("04:19:33.1234")

```

```
msf <- hms::parse_hms("02:55:05")
sjl(msw, msf)
#> [1] "5068.12339997292s (~1.41 hours)" # Expected

round_time(sjl(msw, msf))
#> [1] "5068s (~1.41 hours)" # Expected
```

sjl_sc

Compute Jankowski's MCTQ sleep-corrected social jetlag

Description

[Maturing]

`sjl_sc()` computes the **Jankowski's (2017) sleep-corrected social jetlag** for standard, micro, and shift versions of the Munich ChronoType Questionnaire (MCTQ).

`sjl_sc_rel()` is just a wrapper for `sjl_sc()` with `abs = FALSE`.

Please note that the Jankowski (2017) did not proposed a "relative" sleep-corrected social jetlag, but the user may consider using it.

Usage

```
sjl_sc(so_w, se_w, so_f, se_f, abs = TRUE, method = "shorter")
```

```
sjl_sc_rel(so_w, se_w, so_f, se_f, method = "shorter")
```

Arguments

- | | |
|---------------------|---|
| <code>so_w</code> | An hms object corresponding to the local time of sleep onset on workdays from a standard, micro, or shift version of the MCTQ questionnaire. You can use so() to compute it for the standard or shift version. |
| <code>se_w</code> | An hms object corresponding to the local time of sleep end on workdays from a standard, micro, or shift version of the MCTQ questionnaire. |
| <code>so_f</code> | An hms object corresponding to the local time of sleep onset on work-free days from a standard, micro, or shift version of the MCTQ questionnaire. You can use so() to compute it for the standard or shift version. |
| <code>se_f</code> | An hms object corresponding to the local time of sleep end on work-free days from a standard, micro, or shift version of the MCTQ questionnaire. |
| <code>abs</code> | (optional) a logical object indicating if the function must return an absolute value (default: TRUE). |
| <code>method</code> | (optional) a string indicating which method the function must use to compute the social jetlag. See the Methods section to learn more (default: "shorter"). |

Details

Standard MCTQ functions were created following the guidelines in Roenneberg, Wirz-Justice, & Meroow (2003), Roenneberg, Allebrandt, Meroow, & Vetter (2012), and from The Worldwide Experimental Platform (theWeP, n.d.).

μ **MCTQ** functions were created following the guidelines in Ghotbi et al. (2020), in addition to the guidelines used for the standard MCTQ.

MCTQ^{Shift} functions were created following the guidelines in Juda, Vetter, & Roenneberg (2013), in addition to the guidelines used for the standard MCTQ.

See the References section to learn more.

Class requirements:

The `mctq` package works with a set of object classes specially created to hold time values. These classes can be found in the `lubridate` and `hms` packages. Please refer to those package documentations to learn more about them.

Rounding and fractional time:

Some operations may produce an output with fractional time (e.g., "19538.3828571429s (~5.43 hours)", 01:15:44.505). If you want, you can round it with `round_time()`.

Our recommendation is to avoid rounding, but, if you do, make sure that you only round your values after all computations are done. That way you avoid **round-off errors**.

Value

- If `abs = TRUE`, a `Duration` object corresponding to the absolute sleep-corrected social jetlag.
- If `abs = FALSE`, a `Duration` object corresponding to the relative sleep-corrected social jetlag.

The output may also vary depending on the method used.

Guidelines

In an article published in 2017, Konrad S. Jankowski argued that the original formula for computing the social jetlag (SJL) captures not only the misalignment between social and biological time, but also the sleep debt resulting from sleep deprivation during workdays. Jankowski then proposed the following guideline for a sleep-corrected social jetlag (SJL_{sc}) computation.

Notes:

- The Jankowski's alternative is disputed. We recommend seeing Roenneberg, Pilz, Zerbini, & Winnebeck (2019) discussion about it (see item 3.4.2).
- For **MCTQ^{Shift}**, the computation below must be applied to each shift section of the questionnaire.
- Due to time arithmetic issues, `sjl_sc()` does a slightly different computation by default than those proposed by the author mentioned above. See `vignette("sjl-computation", package = "mctq")` for more details.
- If you are visualizing this documentation in plain text, you may have some trouble understanding the equations. You can see this documentation on the package [website](#).

For standard and micro versions of the MCTQ:

$$\text{If } SD_W > SD_F \ \& \ SE_W \leq SE_F, \ |SE_F - SE_W|$$

$$\text{Else } , \ |SO_F - SO_W|$$

Where:

- SO_W = Local time of sleep onset on workdays.
- SE_W = Local time of sleep end on workdays.
- SO_F = Local time of sleep onset on work-free days.
- SE_F = Local time of sleep end on work-free days.

* W = Workdays; F = Work-free days.

For the shift version of the MCTQ:

$$\text{If } SD_W^{M/E/N} > SD_F^{M/E/N} \ \& \ SE_W^{M/E/N} \leq SE_F^{M/E/N}, \ |SE_F^{M/E/N} - SE_W^{M/E/N}|$$

$$\text{Else } , \ |SO_F^{M/E/N} - SO_W^{M/E/N}|$$

Where:

- $SO_W^{M/E/N}$ = Local time of sleep onset between two days in a particular shift.
- $SE_W^{M/E/N}$ = Local time of sleep end between two days in a particular shift.
- $SO_F^{M/E/N}$ = Local time of sleep onset between two free days after a particular shift.
- $SE_F^{M/E/N}$ = Local time of sleep end between two free days after a particular shift.

* W = Workdays; F = Work-free days, M = Morning shift; E = Evening shift; N = Night shift.

Methods for computing the sleep-corrected social jetlag

There are different approaches to compute the sleep-corrected social jetlag (SJL_{sc}). By default, `sjl_sc()` uses an approach that we call "the shorter interval approach" ("shorter").

The topics below provide a simple explanation of each method supported by `sjl_sc()`. To get a detail understating of this methods, see `vignette("sjl-computation", package = "mctq")`.

- "difference"

By using `method = "difference"`, `sjl_sc()` will do the exact computation proposed by Jankowski, i.e., SJL_{sc} will be computed as the linear difference between SO_f/SE_f and SO_W/SE_W (see the Guidelines section).

We do not recommend using this method, as it has many limitations.

- "shorter"

This is the default method for `sjl_sc()`. It's based on the shorter interval between SO_f/SE_f and SO_W/SE_W , solving most of the issues relating to SJL_{sc} computation.

- "longer"

The "longer" method uses the same logic of the "shorter" method, but, instead of using the shorter interval between SO_f/SE_f and SO_w/SE_w , it uses the longer interval between the two, considering a two-day window.

This method may help in special contexts, like when dealing with shift-workers that have a greater than 12 hours distance between their sleep hours.

References

- Ghotbi, N., Pilz, L. K., Winnebeck, E. C., Vetter, C., Zerbini, G., Lenssen, D., Frighetto, G., Salamanca, M., Costa, R., Montagnese, S., & Roenneberg, T. (2020). The μ MCTQ: an ultra-short version of the Munich ChronoType Questionnaire. *Journal of Biological Rhythms*, 35(1), 98-110. doi: [10.1177/0748730419886986](https://doi.org/10.1177/0748730419886986).
- Jankowski K. S. (2017). Social jet lag: sleep-corrected formula. *Chronobiology International*, 34(4), 531-535. doi: [10.1080/07420528.2017.1299162](https://doi.org/10.1080/07420528.2017.1299162).
- Juda, M., Vetter, C., & Roenneberg, T. (2013). The Munich ChronoType Questionnaire for shift-workers (MCTQ^{Shift}). *Journal of Biological Rhythms*, 28(2), 130-140. doi: [10.1177/0748730412475041](https://doi.org/10.1177/0748730412475041).
- Roenneberg T., Allebrandt K. V., Mewes M., & Vetter C. (2012). Social jetlag and obesity. *Current Biology*, 22(10), 939-43. doi: [10.1016/j.cub.2012.03.038](https://doi.org/10.1016/j.cub.2012.03.038).
- Roenneberg, T., Pilz, L. K., Zerbini, G., & Winnebeck, E. C. (2019). Chronotype and social jetlag: a (self-) critical review. *Biology*, 8(3), 54. doi: [10.3390/biology8030054](https://doi.org/10.3390/biology8030054).
- Roenneberg, T., Wirz-Justice, A., & Mewes, M. (2003). Life between clocks: daily temporal patterns of human chronotypes. *Journal of Biological Rhythms*, 18(1), 80-90. doi: [10.1177/0748730402239679](https://doi.org/10.1177/0748730402239679).
- The Worldwide Experimental Platform (n.d.). MCTQ. <https://www.thewep.org/documentations/mctq/>

See Also

Other MCTQ functions: [fd\(\)](#), [gu\(\)](#), [le_week\(\)](#), [msf_sc\(\)](#), [msl\(\)](#), [napd\(\)](#), [sd24\(\)](#), [sd_overall\(\)](#), [sd_week\(\)](#), [sdu\(\)](#), [sjl_weighted\(\)](#), [sjl\(\)](#), [so\(\)](#), [tbt\(\)](#)

Examples

```
## Scalar example

so_w <- hms::parse_hm("02:00")
se_w <- hms::parse_hm("10:00")
so_f <- hms::parse_hm("01:00")
se_f <- hms::parse_hm("08:00")

sjl_sc(so_w, se_w, so_f, se_f)
#> [1] "3600s (~1 hours)" # Expected
sjl_sc(so_w, se_w, so_f, se_f, abs = FALSE)
#> [1] "-3600s (~-1 hours)" # Expected (negative sjl_sc)
sjl_sc_rel(so_w, se_w, so_f, se_f) # Wrapper function
#> [1] "-3600s (~-1 hours)" # Expected (negative sjl_sc)
```

```

sjl(msl(so_w, sdu(so_w, se_w)), msl(so_f, sdu(so_f, se_f)))
#> [1] "5400s (~1.5 hours)" # Expected

so_w <- hms::parse_hm("22:00")
se_w <- hms::parse_hm("06:00")
so_f <- hms::parse_hm("01:00")
se_f <- hms::parse_hm("06:00") # sd_w > sd_f & se_w <= se_f

sjl_sc(so_w, se_w, so_f, se_f) # sjl_sc = | se_f - se_w |
#> [1] "0s" # Expected
sjl_sc(so_w, se_w, so_f, se_f, abs = FALSE)
#> [1] "0s" # Expected
sjl_sc_rel(so_w, se_w, so_f, se_f) # Wrapper function
#> [1] "0s" # Expected
sjl(msl(so_w, sdu(so_w, se_w)), msl(so_f, sdu(so_f, se_f)))
#> [1] "5400s (~1.5 hours)" # Expected

so_f <- hms::as_hms(NA)

sjl_sc(so_w, se_w, so_f, se_f)
#> [1] NA # Expected

## Vector example

so_w <- c(hms::parse_hm("00:00"), hms::parse_hm("01:00"))
se_w <- c(hms::parse_hm("08:00"), hms::parse_hm("07:00"))
so_f <- c(hms::parse_hm("01:00"), hms::parse_hm("01:00"))
se_f <- c(hms::parse_hm("09:00"), hms::parse_hm("09:00"))

sjl_sc(so_w, se_w, so_f, se_f)
#> [1] "3600s (~1 hours)" "0s" # Expected
sjl_sc(so_w, se_w, so_f, se_f, abs = FALSE)
#> [1] "3600s (~1 hours)" "0s" # Expected
sjl_sc_rel(so_w, se_w, so_f, se_f) # Wrapper function
#> [1] "3600s (~1 hours)" "0s" # Expected
sjl(msl(so_w, sdu(so_w, se_w)), msl(so_f, sdu(so_f, se_f)))
#> [1] "3600s (~1 hours)" "3600s (~1 hours)" # Expected

## See other examples in '?sjl()'

```

sjl_weighted

Compute MCTQ absolute social jetlag across all shifts

Description

[Maturing]

sjl_weighted() computes the **absolute social jetlag across all shifts** for the shift version of the Munich ChronoType Questionnaire (MCTQ).

Usage

```
sjl_weighted(sjl, n_w)
```

Arguments

- sjl** A [list](#) object with [Duration](#) elements corresponding to the **social jetlag in each shift** from a shift version of the MCTQ questionnaire (you can use [sjl\(\)](#) to compute it). sjl elements and values must be paired with n elements and values.
- n_w** A [list](#) object with [integerish integer](#) or [double](#) elements corresponding to the **number of days worked in each shift within a shift cycle** from a shift version of the MCTQ questionnaire. n elements and values must be paired with sjl elements and values.

Details

Standard MCTQ functions were created following the guidelines in Roenneberg, Wirz-Justice, & Meroow (2003), Roenneberg, Allebrandt, Meroow, & Vetter (2012), and from The Worldwide Experimental Platform (theWeP, n.d.).

μ **MCTQ** functions were created following the guidelines in Ghotbi et al. (2020), in addition to the guidelines used for the standard MCTQ.

MCTQ^{Shift} functions were created following the guidelines in Juda, Vetter, & Roenneberg (2013), in addition to the guidelines used for the standard MCTQ.

See the References section to learn more.

Class requirements:

The mctq package works with a set of object classes specially created to hold time values. These classes can be found in the [lubridate](#) and [hms](#) packages. Please refer to those package documents to learn more about them.

Rounding and fractional time:

Some operations may produce an output with fractional time (e.g., "19538.3828571429s (~5.43 hours)", 01:15:44.505). If you want, you can round it with [round_time\(\)](#).

Our recommendation is to avoid rounding, but, if you do, make sure that you only round your values after all computations are done. That way you avoid **round-off errors**.

Value

A [Duration](#) object corresponding to the vectorized weighted mean of sjl with n_w as weights.

Operation

The shift version of the MCTQ was developed for shift-workers rotating through morning-, evening-, and night-shifts, but it also allows adaptations to other shift schedules (Juda, Vetter, & Roenneberg, 2013). For this reason, sjl_weighted() must operate with any shift combination.

Considering the requirement above, sjl_weighted() was developed to only accept [list](#) objects as arguments. For this approach to work, both sjl and n_w arguments must be lists with paired

elements and values, i.e., the first element of `sjl` (e.g., `sjl_m`) must be paired with the first element of `n_w` (e.g., `n_w_m`). The function will do the work of combining them and output a weighted mean.

Guidelines

Juda, Vetter, & Roenneberg (2013) and The Worldwide Experimental Platform (n.d.) guidelines for `sjl_weighted()` ($\emptyset SJJL_{weighted}$) computation are as follows.

Notes:

- The absolute social jetlag across all shifts ($\emptyset SJJL_{weighted}$) is the weighted average of all absolute social jetlags.
- The authors describe an equation for a three-shift schedule, but this may not be your case. That's why this function works a little bit differently (see the Operation section), allowing you to compute a weighted average with any shift combination.
- If you are visualizing this documentation in plain text, you may have some trouble understanding the equations. You can see this documentation on the package [website](#).

Computation:

$$\frac{|SJJL^M| \times n_W^M + |SJJL^E| \times n_W^E + |SJJL^N| \times n_W^N}{n_W^M + n_W^E + n_W^N}$$

Where:

- $SJJL^{M/E/N}$ = Absolute social jetlag in each shift.
- $n_W^{M/E/N}$ = Number of days worked in each shift within a shift cycle.

* W = Workdays; F = Work-free days, M = Morning shift; E = Evening shift; N = Night shift.

References

- Ghotbi, N., Pilz, L. K., Winnebeck, E. C., Vetter, C., Zerbini, G., Lenssen, D., Frighetto, G., Salamanca, M., Costa, R., Montagnese, S., & Roenneberg, T. (2020). The μ MCTQ: an ultra-short version of the Munich ChronoType Questionnaire. *Journal of Biological Rhythms*, 35(1), 98-110. doi: [10.1177/0748730419886986](https://doi.org/10.1177/0748730419886986).
- Juda, M., Vetter, C., & Roenneberg, T. (2013). The Munich ChronoType Questionnaire for shift-workers (MCTQ^{Shift}). *Journal of Biological Rhythms*, 28(2), 130-140. doi: [10.1177/0748730412475041](https://doi.org/10.1177/0748730412475041).
- Roenneberg T., Allebrandt K. V., Mellow M., & Vetter C. (2012). Social jetlag and obesity. *Current Biology*, 22(10), 939-43. doi: [10.1016/j.cub.2012.03.038](https://doi.org/10.1016/j.cub.2012.03.038).
- Roenneberg, T., Wirz-Justice, A., & Mellow, M. (2003). Life between clocks: daily temporal patterns of human chronotypes. *Journal of Biological Rhythms*, 18(1), 80-90. doi: [10.1177/0748730402239679](https://doi.org/10.1177/0748730402239679).
- The Worldwide Experimental Platform (n.d.). MCTQ. <https://www.thewep.org/documentations/mctq/>

See Also

Other MCTQ functions: `fd()`, `gu()`, `le_week()`, `msf_sc()`, `msl()`, `napd()`, `sd24()`, `sd_overall()`, `sd_week()`, `sdu()`, `sjl_sc()`, `sjl()`, `so()`, `tbt()`

Examples

```
## Scalar example

sjl <- list(sjl_m = lubridate::dhours(1.25),
           sjl_e = lubridate::dhours(0.5),
           sjl_n = lubridate::dhours(3))
n_w <- list(n_w_m = 3, n_w_e = 1, n_w_n = 4)
sjl_weighted(sjl, n_w)
#> [1] "7312.5s (~2.03 hours)" # Expected

sjl <- list(sjl_m = lubridate::dhours(1.25),
           sjl_e = lubridate::as.duration(NA),
           sjl_n = lubridate::dhours(3))
n_w <- list(n_w_m = 3, n_w_e = 1, n_w_n = 4)
sjl_weighted(sjl, n_w)
#> [1] NA # Expected

## Vector example

sjl <- list(sjl_m = c(lubridate::dhours(2), lubridate::dhours(2.45)),
           sjl_e = c(lubridate::dhours(3.21), lubridate::as.duration(NA)),
           sjl_n = c(lubridate::dhours(1.2), lubridate::dhours(5.32)))
n_w <- list(n_w_m = c(1, 3), n_w_e = c(4, 1), n_w_n = c(3, 3))
sjl_weighted(sjl, n_w)
#> [1] "8298s (~2.31 hours)" NA # Expected

## Checking the first output from vector example

if (requireNamespace("stats", quietly = TRUE)) {
  i <- 1
  x <- c(sjl[["sjl_m"]][i], sjl[["sjl_e"]][i], sjl[["sjl_n"]][i])
  w <- c(n_w[["n_w_m"]][i], n_w[["n_w_e"]][i], n_w[["n_w_n"]][i])
  lubridate::as.duration(stats::weighted.mean(x, w))
}
#> [1] "8298s (~2.31 hours)" # Expected

## Converting the output to hms

sjl <- list(sjl_m = lubridate::dhours(0.25),
           sjl_e = lubridate::dhours(1.2),
           sjl_n = lubridate::dhours(4.32))
n_w <- list(n_w_m = 4, n_w_e = 2, n_w_n = 1)

sjl_weighted(sjl, n_w)
#> [1] "3970.28571428571s (~1.1 hours)" # Expected

hms::as_hms(as.numeric(sjl_weighted(sjl, n_w)))
#> 01:06:10.285714 # Expected

## Rounding the output at the seconds level

round_time(sjl_weighted(sjl, n_w))
```

```
#> [1] "3970s (~1.1 hours)" # Expected

round_time(hms::as_hms(as.numeric(sjl_weighted(sjl, n_w))))
#> 01:06:10 # Expected
```

sloss_week

Compute MCTQ weekly sleep loss

Description

[Maturing]

sloss_week() computes the **weekly sleep loss** for the standard and micro versions of the Munich ChronoType Questionnaire (MCTQ).

Usage

```
sloss_week(sd_w, sd_f, wd)
```

Arguments

sd_w	A Duration object corresponding to the sleep duration on workdays from a standard or micro version of the MCTQ questionnaire. You can use sdu() to compute it.
sd_f	A Duration object corresponding to the sleep duration on work-free days from a standard or micro version of the MCTQ questionnaire. You can use sdu() to compute it.
wd	An integerish numeric object or an integer object corresponding to the number of workdays per week from a standard or micro version of the MCTQ questionnaire.

Details

Standard MCTQ functions were created following the guidelines in Roenneberg, Wirz-Justice, & Merrow (2003), Roenneberg, Allebrandt, Merrow, & Vetter (2012), and from The Worldwide Experimental Platform (theWeP, n.d.).

μ **MCTQ** functions were created following the guidelines in Ghotbi et al. (2020), in addition to the guidelines used for the standard MCTQ.

MCTQ^{Shift} functions were created following the guidelines in Juda, Vetter, & Roenneberg (2013), in addition to the guidelines used for the standard MCTQ.

See the References section to learn more.

Class requirements:

The mctq package works with a set of object classes specially created to hold time values. These classes can be found in the [lubridate](#) and [hms](#) packages. Please refer to those package documents to learn more about them.

Rounding and fractional time:

Some operations may produce an output with fractional time (e.g., "19538.3828571429s (~5.43 hours)", 01:15:44.505). If you want, you can round it with `round_time()`.

Our recommendation is to avoid rounding, but, if you do, make sure that you only round your values after all computations are done. That way you avoid **round-off errors**.

Value

A `Duration` object corresponding to the weekly sleep loss.

Guidelines

Roenneberg, Allebrandt, Merrow, & Vetter (2012) and The Worldwide Experimental Platform (n.d.) guidelines for `sloss_week()` ($SLoss_{week}$) computation are as follows.

Notes:

- If you are visualizing this documentation in plain text, you may have some trouble understanding the equations. You can see this documentation on the package [website](#).

Computation:

$$\text{If } SD_{week} > SD_W, (SD_{week} - SD_W) \times WD$$

$$\text{If } SD_{week} \leq SD_W, (SD_{week} - SD_F) \times FD$$

Where:

- SD_W = Sleep duration on workdays.
- SD_F = Sleep duration on work-free days.
- SD_{week} = average weekly sleep duration.
- WD = Number of workdays per week ("I have a regular work schedule and work ___ days per week").
- FD = Number of work-free days per week.

* W = Workdays; F = Work-free days.

References

- Ghotbi, N., Pilz, L. K., Winnebeck, E. C., Vetter, C., Zerbini, G., Lenssen, D., Frighetto, G., Salamanca, M., Costa, R., Montagnese, S., & Roenneberg, T. (2020). The μ MCTQ: an ultra-short version of the Munich ChronoType Questionnaire. *Journal of Biological Rhythms*, 35(1), 98-110. doi: [10.1177/0748730419886986](https://doi.org/10.1177/0748730419886986).
- Juda, M., Vetter, C., & Roenneberg, T. (2013). The Munich ChronoType Questionnaire for shift-workers (MCTQ^{Shift}). *Journal of Biological Rhythms*, 28(2), 130-140. doi: [10.1177/0748730412475041](https://doi.org/10.1177/0748730412475041).
- Roenneberg T., Allebrandt K. V., Merrow M., & Vetter C. (2012). Social jetlag and obesity. *Current Biology*, 22(10), 939-43. doi: [10.1016/j.cub.2012.03.038](https://doi.org/10.1016/j.cub.2012.03.038).

Roenneberg, T., Wirz-Justice, A., & Merrow, M. (2003). Life between clocks: daily temporal patterns of human chronotypes. *Journal of Biological Rhythms*, 18(1), 80-90. doi: [10.1177/0748730402239679](https://doi.org/10.1177/0748730402239679).

The Worldwide Experimental Platform (n.d.). MCTQ. <https://www.thewep.org/documentations/mctq/>

Examples

```
## Scalar example

sd_w <- lubridate::dhours(6.5)
sd_f <- lubridate::dhours(7)
wd <- 4
sloss_week(sd_w, sd_f, wd)
#> [1] "3085.71428571429s (~51.43 minutes)" # Expected

sd_w <- lubridate::dhours(7)
sd_f <- lubridate::dhours(8)
wd <- 5
sloss_week(sd_w, sd_f, wd)
#> [1] "5142.85714285714s (~1.43 hours)" # Expected

sd_w <- lubridate::dhours(NA)
sd_f <- lubridate::dhours(9.45)
wd <- 7
sloss_week(sd_w, sd_f, wd)
#> [1] NA # Expected

## Vector example

sd_w <- c(lubridate::dhours(7), lubridate::dhours(8))
sd_f <- c(lubridate::dhours(6.5), lubridate::dhours(8))
wd <- c(2, 0)
sloss_week(sd_w, sd_f, wd)
#> [1] "2571.42857142857s (~42.86 minutes)" "0s" # Expected

## Converting the output to 'hms'

sd_w <- lubridate::dhours(4)
sd_f <- lubridate::dhours(5)
wd <- 3
sloss_week(sd_w, sd_f, wd)
#> [1] "6171.42857142858s (~1.71 hours)" # Expected

hms::as_hms(as.numeric(sloss_week(sd_w, sd_f, wd)))
#> 01:42:51.428571 # Expected

## Rounding the output at the seconds level

sd_w <- lubridate::dhours(5.8743)
sd_f <- lubridate::dhours(7.4324)
wd <- 6
```

```
sloss_week(sd_w, sd_f, wd)
#> [1] "4807.85142857144s (~1.34 hours)" # Expected

round_time(sloss_week(sd_w, sd_f, wd))
#> [1] "4808s (~1.34 hours)" # Expected
```

so

Compute MCTQ local time of sleep onset

Description

[Maturing]

`so()` computes the **local time of sleep onset** for standard and shift versions of the Munich Chrono-Type Questionnaire (MCTQ).

Note that this value is collected directly from the questionnaire if you're using the μ MCTQ.

Usage

```
so(sprep, slat)
```

Arguments

<code>sprep</code>	An hms object corresponding to the local time of preparing to sleep from a standard or shift version of the MCTQ questionnaire.
<code>slat</code>	A Duration object corresponding to the sleep latency or time to fall asleep after preparing to sleep from a standard or shift version of the MCTQ questionnaire.

Details

Standard MCTQ functions were created following the guidelines in Roenneberg, Wirz-Justice, & Mewes (2003), Roenneberg, Allebrandt, Mewes, & Vetter (2012), and from The Worldwide Experimental Platform (theWeP, n.d.).

μ MCTQ functions were created following the guidelines in Ghotbi et al. (2020), in addition to the guidelines used for the standard MCTQ.

MCTQ^{Shift} functions were created following the guidelines in Juda, Vetter, & Roenneberg (2013), in addition to the guidelines used for the standard MCTQ.

See the References section to learn more.

Class requirements:

The `mctq` package works with a set of object classes specially created to hold time values. These classes can be found in the [lubridate](#) and [hms](#) packages. Please refer to those package documentations to learn more about them.

Rounding and fractional time:

Some operations may produce an output with fractional time (e.g., "19538.3828571429s (~5.43 hours)", 01:15:44.505). If you want, you can round it with `round_time()`.

Our recommendation is to avoid rounding, but, if you do, make sure that you only round your values after all computations are done. That way you avoid **round-off errors**.

Value

An `hms` object corresponding to the vectorized sum of `sprep` and `slat` in a circular time frame of 24 hours.

Guidelines

Roenneberg, Allebrandt, Merrow, & Vetter (2012), Juda, Vetter, & Roenneberg (2013), and The Worldwide Experimental Platform (n.d.) guidelines for `so()` (*SO*) computation are as follows.

Notes:

- This computation must be applied to each section of the questionnaire.
- If you are visualizing this documentation in plain text, you may have some trouble understanding the equations. You can see this documentation on the package [website](#).

For standard and micro versions of the MCTQ:

$$SPrep_{W/F} + SLat_{W/F}$$

Where:

- $SPrep_{W/F}$ = Local time of preparing to sleep on work **or** work-free days ("I actually get ready to fall asleep at ___ o'clock").
- $SLat_{W/F}$ = Sleep latency or time to fall asleep after preparing to sleep on work **or** work-free days ("I need ___ min to fall asleep").

* W = Workdays; F = Work-free days.

For the shift version of the MCTQ:

$$SPrep_{W/F}^{M/E/N} + SLat_{W/F}^{M/E/N}$$

Where:

- $SPrep_{W/F}^{M/E/N}$ = Local time of preparing to sleep between two days in a particular shift **or** between two free days after a particular shift ("I actually get ready to fall asleep at ___ o'clock").
- $SLat_{W/F}^{M/E/N}$ = Sleep latency or time to fall asleep after preparing to sleep between two days in a particular shift **or** between two free days after a particular shift ("I need ___ min to fall asleep").

* W = Workdays; F = Work-free days, M = Morning shift; E = Evening shift; N = Night shift.

References

- Ghotbi, N., Pilz, L. K., Winnebeck, E. C., Vetter, C., Zerbini, G., Lenssen, D., Frighetto, G., Salamanca, M., Costa, R., Montagnese, S., & Roenneberg, T. (2020). The μ MCTQ: an ultra-short version of the Munich ChronoType Questionnaire. *Journal of Biological Rhythms*, 35(1), 98-110. doi: [10.1177/0748730419886986](https://doi.org/10.1177/0748730419886986).
- Juda, M., Vetter, C., & Roenneberg, T. (2013). The Munich ChronoType Questionnaire for shift-workers (MCTQ^{Shift}). *Journal of Biological Rhythms*, 28(2), 130-140. doi: [10.1177/0748730412475041](https://doi.org/10.1177/0748730412475041).
- Roenneberg T., Allebrandt K. V., Merrow M., & Vetter C. (2012). Social jetlag and obesity. *Current Biology*, 22(10), 939-43. doi: [10.1016/j.cub.2012.03.038](https://doi.org/10.1016/j.cub.2012.03.038).
- Roenneberg, T., Wirz-Justice, A., & Merrow, M. (2003). Life between clocks: daily temporal patterns of human chronotypes. *Journal of Biological Rhythms*, 18(1), 80-90. doi: [10.1177/0748730402239679](https://doi.org/10.1177/0748730402239679).
- The Worldwide Experimental Platform (n.d.). MCTQ. <https://www.thewep.org/documentations/mctq/>

See Also

Other MCTQ functions: [fd\(\)](#), [gu\(\)](#), [le_week\(\)](#), [msf_sc\(\)](#), [msl\(\)](#), [napd\(\)](#), [sd24\(\)](#), [sd_overall\(\)](#), [sd_week\(\)](#), [sdu\(\)](#), [sjl_sc\(\)](#), [sjl_weighted\(\)](#), [sjl\(\)](#), [tbt\(\)](#)

Examples

```
## Scalar example

sprep <- hms::parse_hm("22:00")
slat <- lubridate::dminutes(15)
so(sprep, slat)
#> 22:15:00 # Expected

sprep <- hms::parse_hm("23:30")
slat <- lubridate::dminutes(45)
so(sprep, slat)
#> 00:15:00 # Expected

sprep <- hms::parse_hm("20:45")
slat <- lubridate::as.duration(NA)
so(sprep, slat)
#> NA # Expected

## Vector example

sprep <- c(hms::parse_hm("21:30"), hms::parse_hm("22:15"))
slat <- c(lubridate::dminutes(45), lubridate::dminutes(5))
so(sprep, slat)
#> 22:15:00 # Expected
#> 22:20:00 # Expected
```

`std_mctq`*A fictional standard MCTQ dataset*

Description

[Maturing]

A fictional dataset, **for testing and learning purposes**, composed of basic/measurable and computed variables of the Munich ChronoType Questionnaire (MCTQ) standard version.

This data was created following the guidelines in Roenneberg, Wirz-Justice, & Merrow (2003), Roenneberg, Allebrandt, Merrow, & Vetter (2012), Jankowski (2017), and The Worldwide Experimental Platform (n.d.). See the References and Details sections to learn more.

Usage

`std_mctq`

Format

A [tibble](#) with 39 columns and 50 rows:

id A unique [integer](#) value to identify each respondent in the dataset.

Type: Control.

R class: [integer](#).

work A [logical](#) value indicating if the respondent has a regular work schedule.

Statement (EN): "I have a regular work schedule (this includes being, for example, a housewife or househusband): Yes (___) No (___)".

Type: Basic.

R class: [logical](#).

wd Number of **workdays** per week.

Statement (EN): "I have a regular work schedule and work ___ days per week".

Type: Basic.

R class: [integer](#).

fd Number of **work-free days** per week.

Type: Computed.

R class: [integer](#).

bt_w Local time of going to bed on **workdays**.

Statement (EN): "I go to bed at ___ o'clock".

Type: Basic.

R class: [hms](#).

sprep_w Local time of preparing to sleep on **workdays**.

Statement (EN): "I actually get ready to fall asleep at ___ o'clock".

Type: Basic.

R class: [hms](#).

slat_w Sleep latency or time to fall asleep after preparing to sleep on **workdays**.

Statement (EN): "I need ___ minutes to fall asleep".

Type: Basic.

R class: [Duration](#).

so_w Local time of sleep onset on **workdays**.

Type: Computed.

R class: [hms](#).

se_w Local time of sleep end on **workdays**.

Statement (EN): "I wake up at ___ o'clock".

Type: Basic.

R class: [hms](#).

si_w "Sleep inertia" on **workdays**.

Despite the name, this variable represents the time the respondent takes to get up after sleep end.

Statement (EN): "After ___ minutes, I get up".

Type: Basic.

R class: [Duration](#).

gu_w Local time of getting out of bed on **workdays**.

Type: Computed.

R class: **hms**.

alarm_w A **logical** value indicating if the respondent uses an alarm clock to wake up on **workdays**.

Statement (EN): "I use an alarm clock on workdays: Yes (___) No (___)".

Type: Basic.

R class: **logical**.

wake_before_w A **logical** value indicating if the respondent regularly wakes up **before** the alarm rings on **workdays**.

Statement (EN): "If "Yes": I regularly wake up BEFORE the alarm rings: Yes (___) No (___)".

Type: Basic.

R class: **logical**.

sd_w Sleep duration on **workdays**.

Type: Computed.

R class: **Duration**.

tbt_w Total time in bed on **workdays**.

Type: Computed.

R class: **Duration**.

le_w Light exposure on **workdays**.

Statement (EN): "On average, I spend the following amount of time outdoors in daylight (without a roof above my head)".

Type: Extra.

R class: **Duration**.

msw Local time of mid-sleep on **workdays**.

Type: Computed.

R class: **hms**.

bt_f Local time of going to bed on **work-free days**.

Statement (EN): "I go to bed at ___ o'clock".

Type: Basic.

R class: *hms*.

sprep_f Local time of preparing to sleep on **work-free days**.

Statement (EN): "I actually get ready to fall asleep at ___ o'clock".

Type: Basic.

R class: *hms*.

slat_f Sleep latency or time to fall asleep after preparing to sleep on **work-free days**.

Statement (EN): "I need ___ minutes to fall asleep".

Type: Basic.

R class: *Duration*.

so_f Local time of sleep onset on **work-free days**.

Type: Computed.

R class: *hms*.

se_f Local time of sleep end on **work-free days**.

Statement (EN): "I wake up at ___ o'clock".

Type: Basic.

R class: *hms*.

si_f "Sleep inertia" on **work-free days**.

Despite the name, this variable represents the time the respondent takes to get up after sleep end.

Statement (EN): "After ___ minutes, I get up".

Type: Basic.

R class: *Duration*.

gu_f Local time of getting out of bed on **work-free days**.

Type: Computed.

R class: *hms*.

alarm_f A *logical* value indicating if the respondent uses an alarm clock to wake up on **work-free days**.

Statement (EN): "My wake-up time is due to the use of an alarm clock: Yes (___) No (

___)".

Type: Basic.

R class: [logical](#).

reasons_f A [logical](#) value indicating if the respondent has any particular reasons for why they **cannot** freely choose their sleep times on **work-free days**.

Statement (EN): "There are particular reasons why I **cannot** freely choose my sleep times on free days: Yes (___) No (___)".

Type: Basic.

R class: [logical](#).

reasons_why_f Particular reasons for why the respondent cannot freely choose their sleep times on **work-free days**.

Statement (EN): "If "Yes": Child(ren)/pet(s) (___) Hobbies (___) Others (___), for example: ___".

Type: Basic.

R class: character.

sd_f Sleep duration on **work-free days**.

Type: Computed.

R class: [Duration](#).

tbt_f Total time in bed on **work-free days**.

Type: Computed.

R class: [Duration](#).

le_f Light exposure on **work-free days**.

Statement (EN): "On average, I spend the following amount of time outdoors in daylight (without a roof above my head)".

Type: Extra.

R class: [Duration](#).

msf Local time of mid-sleep on **work-free days**.

Type: Computed.

R class: [hms](#).

sd_week Average weekly sleep duration.

Type: Computed.

R class: [Duration](#).

sloss_week Weekly sleep loss.

Type: Computed.

R class: [Duration](#).

le_week Average weekly light exposure.

Type: Computed.

R class: [Duration](#).

msf_sc Chronotype or corrected local time of mid-sleep on **work-free days**.

Type: Computed.

R class: [hms](#).

sjl_rel Relative social jetlag.

Type: Computed.

R class: [Duration](#).

sjl Absolute social jetlag.

Type: Computed.

R class: [Duration](#).

sjl_sc_rel Jankowski's relative sleep-corrected social jetlag.

Type: Computed.

R class: [Duration](#).

sjl_sc Jankowski's sleep-corrected social jetlag.

Type: Computed.

R class: [Duration](#).

Details

std_mctq is a tidied, validated, and transformed version of `raw_data("std_mctq.csv")`.

Guidelines:

To learn more about the Munich ChronoType Questionnaire (MCTQ), see Roenneberg, Wirz-Justice, & Mellow (2003), Roenneberg, Allebrandt, Mellow, & Vetter (2012), Roenneberg et al. (2015), and Roenneberg, Pilz, Zerbini, & Winnebeck (2019).

To know about different MCTQ versions, see Juda, Vetter, & Roenneberg (2013) and Ghotbi et al. (2020).

To learn about the sleep-corrected social jetlag, see Jankowski (2017).

If you're curious about the variable computations and want to have access to the full questionnaire, see The Worldwide Experimental Platform (n.d.).

Data building and data wrangling:

This dataset was created by randomized sampling (see `random_mctq()`) and by manual insertions of special cases. Its purpose is to demonstrate common cases and data issues that researchers may find in their MCTQ data, in addition to be a suggested data structure for MCTQ data.

You can see the `std_mctq` build and data wrangling processes [here](#).

Variable naming:

The naming of the variables took into account the naming scheme used in MCTQ publications, in addition to the guidelines of the [tidyverse style guide](#).

Variable classes:

The `mctq` package works with a set of object classes specially created to hold time values. These classes can be found in the [hms](#) and [lubridate](#) package.

Duration objects:

If you prefer to view `Duration` objects as `hms` objects, run `pretty_mctq(std_mctq)`.

Source

Created by Daniel Vartanian (package author).

References

- Ghotbi, N., Pilz, L. K., Winnebeck, E. C., Vetter, C., Zerbini, G., Lenssen, D., Frighetto, G., Salamanca, M., Costa, R., Montagnese, S., & Roenneberg, T. (2020). The μ MCTQ: an ultra-short version of the Munich ChronoType Questionnaire. *Journal of Biological Rhythms*, 35(1), 98-110. doi: [10.1177/0748730419886986](https://doi.org/10.1177/0748730419886986).
- Jankowski K. S. (2017). Social jet lag: sleep-corrected formula. *Chronobiology International*, 34(4), 531-535. doi: [10.1080/07420528.2017.1299162](https://doi.org/10.1080/07420528.2017.1299162).
- Juda, M., Vetter, C., & Roenneberg, T. (2013). The Munich ChronoType Questionnaire for shift-workers (MCTQ^{Shift}). *Journal of Biological Rhythms*, 28(2), 130-140. doi: [10.1177/0748730412475041](https://doi.org/10.1177/0748730412475041).
- Roenneberg T., Allebrandt K. V., Mellow M., & Vetter C. (2012). Social jetlag and obesity. *Current Biology*, 22(10), 939-43. doi: [10.1016/j.cub.2012.03.038](https://doi.org/10.1016/j.cub.2012.03.038).
- Roenneberg, T., Keller, L. K., Fischer, D., Matera, J. L., Vetter, C., & Winnebeck, E. C. (2015). Human activity and rest in situ. In A. Sehgal (Ed.), *Methods in Enzymology* (Vol. 552, pp. 257-283). London, UK: Academic Press. doi: [10.1016/bs.mie.2014.11.028](https://doi.org/10.1016/bs.mie.2014.11.028).
- Roenneberg, T., Pilz, L. K., Zerbini, G., & Winnebeck, E. C. (2019). Chronotype and social jetlag: a (self-) critical review. *Biology*, 8(3), 54. doi: [10.3390/biology8030054](https://doi.org/10.3390/biology8030054).

Roenneberg, T., Wirz-Justice, A., & Merrow, M. (2003). Life between clocks: daily temporal patterns of human chronotypes. *Journal of Biological Rhythms*, 18(1), 80-90. doi: [10.1177/0748730402239679](https://doi.org/10.1177/0748730402239679).

The Worldwide Experimental Platform (n.d.). MCTQ. <https://www.thewep.org/documentations/mctq/>

See Also

Other datasets: [micro_mctq](#), [shift_mctq](#)

sum_time	<i>Sum time objects</i>
----------	-------------------------

Description

[Maturing]

sum_time() returns the sum of the time from different kinds of date/time objects.

vct_sum_time() returns the vectorized sum of the time from different kinds of date/time objects.

Both functions can be set to work with a circular time frame (see Details to learn more).

Usage

```
sum_time(..., cycle = NULL, reverse = TRUE, na_rm = FALSE)
```

```
vct_sum_time(..., cycle = NULL, reverse = TRUE, na_rm = FALSE)
```

Arguments

...	Objects belonging to one of the following classes: Duration , difftime , or hms , POSIXct , POSIXlt , or Interval .
cycle	(optional) A numeric or Duration object of length 1, equal or greater than 0, indicating the cycle length in seconds. If NULL the function will perform a linear sum (see Details to learn more) (default: NULL).
reverse	(optional) A logical value indicating if the function must use a reverse cycle for negative sums (see Details to learn more) (default: TRUE).
na_rm	(optional) a logical value indicating if the function must remove NA values while performing the sum (default: FALSE).

Details

sum_time() **versus** vct_sum_time():

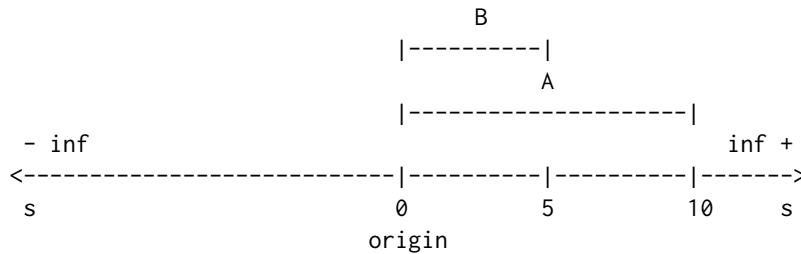
sum_time() behaves similar to [sum\(\)](#), in the sense that it aggregates the time lengths of values in ... into a single data point. For example, sum_time(c(x, y), z) will have the same output as sum_time(x, y, z).

vct_sum_time() performs a different type of sum (a vectorized one). Instead of aggregating the time lengths, the function perform a paired sum between elements. For example, sum_time(c(x, y), c(w, z)) will return a vector like c(sum_time(x, w), sum_time(y, z)). Because of that, vct_sum_time() requires that all objects in ... have the same length.

Linear versus circular time:

Time can have different "shapes".

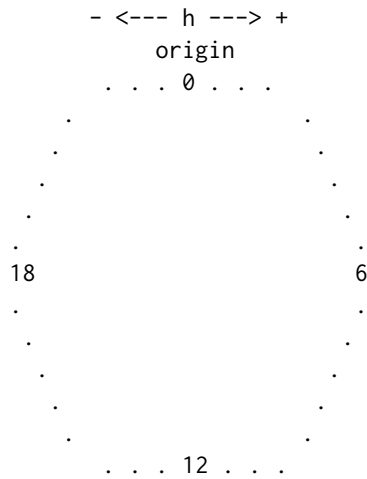
If the objective is to measure the duration (time span) of an event, time is usually measured considering a linear frame, with a fixed point of origin. In this context, the time value distance itself to infinity in relation to the origin.



$$A + B = 10 + 5 = 15s$$

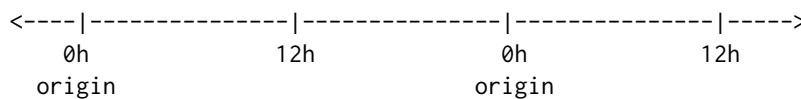
But that's not the only possible "shape" of time, as it can also be measured in other contexts.

In a "time of day" context, time will be linked to the rotation of the earth, "resetting" when a new rotation cycle starts. That brings a different kind of shape to time: a circular shape. With this shape the time value encounters the origin at the end of each cycle.



$$18 + 6 = 0h$$

If we transpose this circular time frame to a linear one, it would look like this:



Note that now the origin is not fix, but cyclical.

`sum_time()` and `vct_sum_time()` can both operate in either a linear or a circular fashion. If `cycle = NULL` (default), the function will use a linear approach. Else, the function will use a circular approach relative to the cycle length (e.g, `cycle = 86400` (1 day)).

Fractional time:

`sum_time()` uses the `%` operator to cycle values. Hence, it can be subject to catastrophic loss of accuracy if values in `...` are fractional and much larger than `cycle`. A warning is given if this is detected.

`%` is a builtin R function that operates like this:

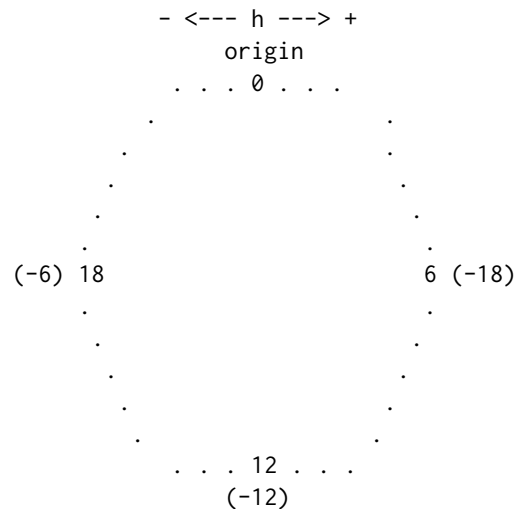
```
function(a, b) {
  a - floor(a / b) * b
}
```

Negative time cycling:

If the sum of the time is negative, with a `cycle` assigned and `reverse = FALSE`, `sum_time()` and `vct_sum_time()` will perform the cycle considering the absolute value of the sum and return the result with a negative signal.

However, If the sum of the time have a negative value, with a `cycle` assigned and `reverse = TRUE` (default), `sum_time()` and `vct_sum_time()` will perform the cycle in reverse, relative to its origin.

Example: If the sum of the time have a -30h time span in a reversed cycle of 24h, the result will be 18h. By removing the full cycles of -30h you will get -6h (-30 + 24), and -6h relative to the origin will be 18h.



Period objects:

`Period` objects are special type of objects developed by the `lubridate` team that represents "human units", ignoring possible timeline irregularities. That is to say that 1 day as `Period` can have different time spans, when looking to a timeline after a irregularity event.

Since the time span of a `Period` object can fluctuate, `sum_time()` and `vct_sum_time()` don't accept this kind of object. You can transform it to a `Duration` object and still use the functions, but beware that this can produce errors.

Learn more about [Period](#) objects in the [Dates and times](#) chapter of Wickham & Grolemund book (n.d.).

POSIXt objects:

[POSIXt](#) objects in ... will be stripped of their dates. Only the time will be considered.

Both [POSIXct](#) and [POSIXlt](#) are objects that inherits the class [POSIXt](#). Learn more about it in [?DateTimeClasses](#).

Interval objects:

By using [Interval](#) objects in ..., `sum_time()` and `vct_sum_time()` will consider only their time spans. That is, the amount of seconds of the intervals.

Learn more about [Interval](#) objects in the [Dates and times](#) chapter of Wickham & Grolemund (n.d.).

Timeline irregularities:

This function does not take into account timeline irregularities (e.g., leap years, DST, leap seconds). This may not be an issue for most people, but it must be considered when doing time arithmetic.

Value

- If `cycle = NULL`, a [Duration](#) object with a linear sum of the time from objects in ...
- If `cycle != NULL`, a [Duration](#) object with a circular sum of the time from objects in ...

References

Wickham, H., & Grolemund, G. (n.d.). *R for data science*. Sebastopol, CA: O'Reilly Media. <https://r4ds.had.co.nz>

See Also

Other utility functions: [assign_date\(\)](#), [cycle_time\(\)](#), [pretty_mctq\(\)](#), [qplot_walk\(\)](#), [random_mctq\(\)](#), [raw_data\(\)](#), [round_time\(\)](#), [shorter_interval\(\)](#)

Examples

```
## Non-vectorized sum in an linear time frame

x <- c(as.POSIXct("2020-01-01 15:00:00"), as.POSIXct("1999-05-04 17:30:00"))
y <- lubridate::as.interval(lubridate::dhours(7), as.Date("1970-05-08"))
sum_time(x, y)
#> [1] "1422200s (~1.65 days)" # 39:30:00 # Expected

## Non-vectorized sum in a circular time frame of 24 hours

x <- c(lubridate::dhours(25), lubridate::dhours(5), lubridate::dminutes(50))
sum_time(x, cycle = lubridate::ddays())
#> [1] "24600s (~6.83 hours)" # 06:50:00 # Expected

x <- c(hms::parse_hm("00:15"), hms::parse_hm("02:30"), hms::as_hms(NA))
```

```

sum_time(x, cycle = lubridate::ddays())
#> NA # Expected
sum_time(x, cycle = lubridate::ddays(), na_rm = TRUE)
#> [1] "9900s (~2.75 hours)" # 02:45:00 # Expected

x <- c(lubridate::dhours(-12), lubridate::dhours(-13))
sum_time(x, cycle = lubridate::ddays(), reverse = FALSE)
#> [1] "-3600s (~-1 hours)" # -01:00:00 # Expected

x <- c(lubridate::dhours(-12), lubridate::dhours(-13))
sum_time(x, cycle = lubridate::ddays(), reverse = TRUE)
#> [1] "82800s (~23 hours)" # 23:00:00 # Expected

## Vectorized sum in an linear time frame

x <- c(lubridate::dhours(6), NA)
y <- c(hms::parse_hm("23:00"), hms::parse_hm("10:00"))
vct_sum_time(x, y)
#> [1] "104400s (~1.21 days)" NA # 29:00:00 NA # Expected
vct_sum_time(x, y, na_rm = TRUE)
#> [1] "104400s (~1.21 days)" "36000s (~10 hours)" # Expected

## Vectorized sum in a circular time frame of 24 hours

x <- c(lubridate::dhours(6), NA)
y <- c(hms::parse_hm("23:00"), hms::parse_hm("10:00"))
vct_sum_time(x, y, cycle = lubridate::ddays())
#> [1] "18000s (~5 hours)" NA # Expected
vct_sum_time(x, y, cycle = lubridate::ddays(), na_rm = TRUE)
#> [1] "18000s (~5 hours)" "36000s (~10 hours)" # Expected

x <- c(lubridate::dhours(-49), lubridate::dhours(-24))
y <- c(hms::parse_hm("24:00"), - hms::parse_hm("06:00"))
vct_sum_time(x, y, cycle = lubridate::ddays(), reverse = FALSE)
#> [1] "-3600s (~-1 hours)" "-21600s (~-6 hours)" # Expected

x <- c(lubridate::dhours(-49), lubridate::dhours(-24))
y <- c(hms::parse_hm("24:00"), - hms::parse_hm("06:00"))
vct_sum_time(x, y, cycle = lubridate::ddays(), reverse = TRUE)
#> [1] "82800s (~23 hours)" "64800s (~18 hours)" # Expected

```

tbt

Compute MCTQ total time in bed

Description

[Maturing]

tbt() computes the **total time in bed** for standard and shift versions of the Munich ChronoType Questionnaire (MCTQ).

Usage

```
tbt(bt, gu)
```

Arguments

bt An [hms](#) object corresponding to the **local time of going to bed** from a standard or shift version of the MCTQ questionnaire.

gu An [hms](#) object corresponding to the **local time of getting out of bed** from a standard or shift version of the MCTQ questionnaire. You can use [gu\(\)](#) to compute it.

Details

Standard MCTQ functions were created following the guidelines in Roenneberg, Wirz-Justice, & Mellow (2003), Roenneberg, Allebrandt, Mellow, & Vetter (2012), and from The Worldwide Experimental Platform (theWeP, n.d.).

μ **MCTQ** functions were created following the guidelines in Ghotbi et al. (2020), in addition to the guidelines used for the standard MCTQ.

MCTQ^{Shift} functions were created following the guidelines in Juda, Vetter, & Roenneberg (2013), in addition to the guidelines used for the standard MCTQ.

See the References section to learn more.

Class requirements:

The `mctq` package works with a set of object classes specially created to hold time values. These classes can be found in the [lubridate](#) and [hms](#) packages. Please refer to those package documentations to learn more about them.

Rounding and fractional time:

Some operations may produce an output with fractional time (e.g., "19538.3828571429s (~5.43 hours)", 01:15:44.505). If you want, you can round it with [round_time\(\)](#).

Our recommendation is to avoid rounding, but, if you do, make sure that you only round your values after all computations are done. That way you avoid **round-off errors**.

Value

A [Duration](#) object corresponding to the vectorized difference between `gu` and `bt` in a circular time frame of 24 hours.

Guidelines

Roenneberg, Allebrandt, Mellow, & Vetter (2012), Juda, Vetter, & Roenneberg (2013), and The Worldwide Experimental Platform (n.d.) guidelines for `tbt()` (*TBT*) computation are as follows.

Notes:

- This computation must be applied to each section of the questionnaire.
- If you are visualizing this documentation in plain text, you may have some trouble understanding the equations. You can see this documentation on the package [website](#).

For standard and micro versions of the MCTQ:

$$GU_{W/F} - BT_{W/F}$$

Where:

- $BT_{W/F}$ = Local time of going to bed on work **or** work-free days ("I go to bed at ___ o'clock").
- $GU_{W/F}$ = Local time of getting out of bed on work **or** work-free days.

* W = Workdays; F = Work-free days.

For the shift version of the MCTQ:

$$GU_{W/F}^{M/E/N} - BT_{W/F}^{M/E/N}$$

Where:

- $BT_{W/F}^{M/E/N}$ = Local time of going to bed between two days in a particular shift **or** between two free days after a particular shift ("I go to bed at ___ o'clock").
- $GU_{W/F}^{M/E/N}$ = Local time of getting out of bed between two days in a particular shift **or** between two free days after a particular shift.

* W = Workdays; F = Work-free days, M = Morning shift; E = Evening shift; N = Night shift.

References

- Ghotbi, N., Pilz, L. K., Winnebeck, E. C., Vetter, C., Zerbini, G., Lenssen, D., Frighetto, G., Salamanca, M., Costa, R., Montagnese, S., & Roenneberg, T. (2020). The μ MCTQ: an ultra-short version of the Munich ChronoType Questionnaire. *Journal of Biological Rhythms*, 35(1), 98-110. doi: [10.1177/0748730419886986](https://doi.org/10.1177/0748730419886986).
- Juda, M., Vetter, C., & Roenneberg, T. (2013). The Munich ChronoType Questionnaire for shift-workers (MCTQ^{Shift}). *Journal of Biological Rhythms*, 28(2), 130-140. doi: [10.1177/0748730412475041](https://doi.org/10.1177/0748730412475041).
- Roenneberg T., Allebrandt K. V., Mewes M., & Vetter C. (2012). Social jetlag and obesity. *Current Biology*, 22(10), 939-43. doi: [10.1016/j.cub.2012.03.038](https://doi.org/10.1016/j.cub.2012.03.038).
- Roenneberg, T., Wirz-Justice, A., & Mewes, M. (2003). Life between clocks: daily temporal patterns of human chronotypes. *Journal of Biological Rhythms*, 18(1), 80-90. doi: [10.1177/0748730402239679](https://doi.org/10.1177/0748730402239679).
- The Worldwide Experimental Platform (n.d.). MCTQ. <https://www.thewep.org/documentations/mctq/>

See Also

Other MCTQ functions: `fd()`, `gu()`, `le_week()`, `msf_sc()`, `msl()`, `napd()`, `sd24()`, `sd_overall()`, `sd_week()`, `sdu()`, `sjl_sc()`, `sjl_weighted()`, `sjl()`, `so()`

Examples

```
## Scalar example

bt <- hms::parse_hm("22:10")
gu <- hms::parse_hm("06:15")
tbt(bt, gu)
#> [1] "29100s (~8.08 hours)" # Expected

bt <- hms::parse_hm("01:20")
gu <- hms::parse_hm("14:00")
tbt(bt, gu)
#> [1] "45600s (~12.67 hours)" # Expected

bt <- hms::as_hms(NA)
gu <- hms::parse_hm("07:20")
tbt(bt, gu)
#> [1] NA # Expected

## Vector example

bt <- c(hms::parse_hm("23:50"), hms::parse_hm("02:30"))
gu <- c(hms::parse_hm("09:30"), hms::parse_hm("11:25"))
tbt(bt, gu)
#> [1] "34800s (~9.67 hours)" "32100s (~8.92 hours)" # Expected
```

Index

* MCTQ functions

fd, 8
gu, 10
le_week, 12
msf_sc, 20
msl, 24
napd, 27
sd24, 38
sd_overall, 44
sd_week, 47
sdu, 41
sjl, 62
sjl_sc, 67
sjl_weighted, 71
so, 78
tbt, 92

* datasets

micro_mctq, 16
shift_mctq, 50
std_mctq, 81

* utility functions

assign_date, 3
cycle_time, 5
pretty_mctq, 30
qplot_walk, 31
random_mctq, 34
raw_data, 35
round_time, 36
shorter_interval, 59
sum_time, 88

?DateTimeClasses, 4, 60, 91

?micro_mctq, 34

?round, 37

?shift_mctq, 34

?std_mctq, 34

?timezone, 4, 60

%%, 6, 90

assign_date, 3, 7, 30, 33, 35–37, 61, 91

atomic, 31

base::logical(), 16

base::round(), 37

character, 31, 32, 35, 53, 55

chronotype (msf_sc), 20

cycle_time, 4, 5, 30, 33, 35–37, 61, 91

data.frame, 30, 31

difftime, 5, 30, 32, 36, 88

double, 72

Duration, 5, 7, 10, 13, 17–20, 25, 28, 30, 32,
36–39, 42, 44, 45, 47, 48, 51–59, 61,
63, 68, 72, 75, 76, 78, 82–88, 90, 91,
93

fd, 8, 12, 14, 23, 26, 29, 40, 43, 46, 49, 65, 70,
73, 80, 94

ggplot2, 32

grDevices, 32

gu, 9, 10, 14, 23, 26, 29, 40, 43, 46, 49, 65, 70,
73, 80, 94

gu(), 93

hms, 3, 5, 10, 11, 13, 17–21, 25, 27, 28, 30, 32,
36, 39, 41, 42, 44, 48, 51–60, 62, 63,
67, 68, 72, 75, 78, 79, 82–88, 93

integer, 8, 9, 13, 16, 44, 48, 51, 54, 72, 75, 81

integerish, 8, 13, 44, 48, 72, 75

Interval, 4, 61, 88, 91

le_week, 9, 12, 12, 23, 26, 29, 40, 43, 46, 49,
65, 70, 73, 80, 94

list, 34, 72

logical, 5, 16, 21, 30, 32, 38, 52, 53, 55, 56,
62, 67, 81, 83–85, 88

longer_duration (shorter_interval), 59

longer_interval (shorter_interval), 59

lubridate, 3, 7, 11, 13, 19, 21, 25, 28, 37, 39,
42, 44, 48, 58, 60, 63, 68, 72, 75, 78,
87, 90, 93

micro_mctq, 16, 59, 88
 ms(), 24
 msf (msl), 24
 msf_sc, 9, 12, 14, 20, 26, 29, 40, 43, 46, 49, 65, 70, 73, 80, 94
 msl, 9, 12, 14, 23, 24, 29, 40, 43, 46, 49, 65, 70, 73, 80, 94
 msl(), 20, 41, 62
 msw (msl), 24

 napd, 9, 12, 14, 23, 26, 27, 40, 43, 46, 49, 65, 70, 73, 80, 94
 napd(), 38
 numeric, 3, 5, 8, 13, 44, 48, 75, 88

 Period, 7, 32, 37, 90, 91
 POSIXct, 4, 36, 60, 88, 91
 POSIXct', 32
 POSIXlt, 4, 36, 60, 88, 91
 POSIXt, 3, 4, 59, 60, 91
 pretty_mctq, 4, 7, 30, 33, 35–37, 61, 91
 pretty_mctq(micro_mctq), 19
 pretty_mctq(shift_mctq), 58
 pretty_mctq(std_mctq), 87

 qqplot(), 31, 32
 qqplot_walk, 4, 7, 30, 31, 35–37, 61, 91

 random_mctq, 4, 7, 30, 33, 34, 36, 37, 61, 91
 random_mctq(), 19, 58, 87
 raw_data, 4, 7, 30, 33, 35, 35, 37, 61, 91
 round(), 30
 round_date(), 37
 round_hms(), 37
 round_time, 4, 7, 30, 33, 35, 36, 36, 61, 91
 round_time(), 11, 13, 21, 25, 28, 30, 39, 42, 45, 48, 63, 68, 72, 76, 79, 93

 sd(), 41
 sd24, 9, 12, 14, 23, 26, 29, 38, 43, 46, 49, 65, 70, 73, 80, 94
 sd_overall, 9, 12, 14, 23, 26, 29, 40, 43, 44, 49, 65, 70, 73, 80, 94
 sd_overall(), 20, 47
 sd_week, 9, 12, 14, 23, 26, 29, 40, 43, 46, 47, 65, 70, 73, 80, 94
 sd_week(), 20, 44
 sdu, 9, 12, 14, 23, 26, 29, 40, 41, 46, 49, 65, 70, 73, 80, 94

 sdu(), 20, 24, 25, 38, 44, 47, 48, 75
 shift_mctq, 20, 50, 88
 shorter_duration (shorter_interval), 59
 shorter_interval, 4, 7, 30, 33, 35–37, 59, 91
 sjl, 9, 12, 14, 23, 26, 29, 40, 43, 46, 49, 62, 70, 73, 80, 94
 sjl(), 72
 sjl_rel (sjl), 62
 sjl_sc, 9, 12, 14, 23, 26, 29, 40, 43, 46, 49, 65, 67, 73, 80, 94
 sjl_sc_rel (sjl_sc), 67
 sjl_weighted, 9, 12, 14, 23, 26, 29, 40, 43, 46, 49, 65, 70, 71, 80, 94
 sloss_week, 75
 so, 9, 12, 14, 23, 26, 29, 40, 43, 46, 49, 65, 70, 73, 78, 94
 so(), 25, 41, 67
 stats, 34
 std_mctq, 20, 59, 81
 sum(), 88
 sum_time, 4, 7, 30, 33, 35–37, 61, 88

 tbt, 9, 12, 14, 23, 26, 29, 40, 43, 46, 49, 65, 70, 73, 80, 92
 tibble, 16, 51, 81
 trunc_hms(), 37

 utils, 32

 vct_sum_time (sum_time), 88