Package 'actibase'

October 21, 2026

Type Package

Title Baseline Functions for Actigraphy and Activity Processing and Analysis

Version 0.8.0

Description Provides baseline functions for actigraphy and activity data. This package is intended to be extended by downstream overlays such as 'actiread', 'actimetrics', and 'stepcount'.

License GPL-3

Depends R (>= 4.1.0)

Suggests testthat (>= 3.0.0), utils, covr, knitr, rmarkdown, readr

Encoding UTF-8

LazyData true

Imports magrittr, dplyr, lubridate (>= 1.9.0), hms, tidyr, assertthat, vctrs

URL https://github.com/jhuwit/actibase, https://jhuwit.github.io/actibase/

BugReports https://github.com/jhuwit/actibase/issues

Config/roxygen2/version 8.0.0

Config/testthat/edition 3

VignetteBuilder knitr

NeedsCompilation no

Author John Muschelli [aut, cre] (ORCID: <https://orcid.org/0000-0001-6469-1750>)

Maintainer John Muschelli <muschellij2@gmail.com>

Repository CRAN

Date/Publication 2026-10-21 08:40:16 UTC

Contents

acti_day_index . . . 2
acti_day_start . . . 3
acti_fill_zeros . . . 4
acti_flag_adjacent_invalid_periods . . . 5
acti_flag_cdc_labels . . . 6
acti_flag_contiguous_device_limit_runs . . . 7
acti_flag_device_limit_windows . . . 8
acti_flag_qc . . . 8
acti_flag_repeated_samples . . . 10
acti_flag_spike . . . 10
acti_hms_to_minute . . . 13
acti_minute_to_hms . . . 14
acti_raw_data . . . 14
acti_repeat_time_of_day . . . 15
acti_resample . . . 16
acti_separate_time . . . 17
acti_split_intervals_at_midnight . . . 17
acti_standardize_data . . . 18
acti_standardize_epoch . . . 19
acti_tidy_axes . . . 21
acti_time_of_day . . . 21
acti_time_to_minute . . . 22
as_convert_safe . . . 23
create_day_inclusion . . . 23
get_dynamic_range . . . 24
get_sample_rate . . . 25
get_transformations . . . 25
is.AccData . . . 26
mark_condition . . . 26
strip_hour_shift . . . 27
xyz . . . 27

Index **29**

`acti_day_index` *Calculate a Baseline-Relative Activity-Day Index*

Description

Calculate a Baseline-Relative Activity-Day Index

Usage

```
acti_day_index(
  time,
  day_start = hms::hms(0),
  origin = NULL,
  start = 1L,
  timezone = NULL
)
```

Arguments

time	A POSIXt vector.
day_start	An hms or HH:MM[:SS] day boundary.
origin	Optional Date or POSIXt baseline; by default the first day.
start	The index assigned to the baseline day, either 0 or 1.
timezone	Optional Olson timezone used to define the day boundary.

Value

An integer vector.

Note

A stable recording-relative day number supports faceting and alignment across participants even when their calendar start dates differ.

Examples

```
time = as.POSIXct(c("2020-01-01 03:00:00", "2020-01-01 05:00:00"), tz = "UTC")
acti_day_index(time, day_start = "04:00")
```

---

acti_day_start *Find the Start of an Activity Day*

Description

This is a custom-day-boundary alternative to lubridate::floor_date() with unit = "day". Use floor_date() when days begin at midnight; use acti_day_start() when a recording day begins at a different time, such as 04:00.

Usage

```
acti_day_start(time, day_start = hms::hms(0), timezone = NULL)
```

Arguments

time	A POSIXt vector.
day_start	An hms or HH:MM[:SS] day boundary.
timezone	Optional Olson timezone used to define the day boundary.

Value

A POSIXct vector.

Note

Activity studies often use a non-midnight boundary (for example, 04:00) so nocturnal behavior is assigned to the preceding activity day.

Examples

```
time = as.POSIXct(c("2020-01-01 03:00:00", "2020-01-01 05:00:00"), tz = "UTC")

# With a midnight boundary, this agrees with floor_date().
acti_day_start(time)
lubridate::floor_date(time, unit = "day")

# With a 04:00 boundary, 03:00 belongs to the preceding activity day.
acti_day_start(time, day_start = "04:00")
```

---

acti_fill_zeros *Fill in Zeros*

Description

Fill in Zeros

Usage

```
acti_fill_zeros(data, method = c("all", "actilife"), sample_rate = NULL)

acti_fill_zeroes(data, method = c("all", "actilife"), sample_rate = NULL)

acti_remove_leading_zeros(data)

acti_remove_leading_zeroes(data)
```

Arguments

`data`	a data frame containing columns time, X, Y, Z
`method`	`"all"` (default) fills every all-zero row. `"actilife"` fills only runs of all-zero rows that are at least one second long (`sample_rate` rows) and keeps shorter runs as recorded zeros. In idle sleep the device stops sampling for at least a second, and ActiLife repeats the last sample over that time, but a single recorded `(0, 0, 0)` sample is kept by ActiLife.
`sample_rate`	sample rate, in Hz, used when `method = "actilife"`. If `NULL`, it is resolved with `get_sample_rate()`.

Value

the modified data frame with zeros replaced by NA

Note

With `method = "actilife"`, the kept zero rows are still all zero, so a later step that treats all-zero rows as idle sleep (such as `agcounts::calculate_counts()`) will fill them anyway.

Examples

```
acti_fill_zeros(acti_raw_data)
acti_fill_zeroes(acti_raw_data)
acti_fill_zeros(acti_raw_data, method = "actilife")
```

`acti_flag_adjacent_invalid_periods` *Flag periods adjacent to invalid periods*

Description

Marks the previous and next observed periods when their timestamps are exactly `window_seconds` apart. Groups are evaluated independently.

Usage

```
acti_flag_adjacent_invalid_periods(
  data,
  invalid,
  time = "time",
  window_seconds,
  group = NULL
)
```

Arguments

data	Minute-level summary data.
invalid	Logical vector identifying invalid periods.
time	Name of the timestamp column.
window_seconds	Expected duration between adjacent periods.
group	Optional grouping column name(s).

Value

Input data with flag_adjacent_invalid.

---

acti_flag_cdc_labels *Label quality flags with CDC NHANES letters*

Description

Creates a character column containing the CDC letter codes corresponding to TRUE package flag columns. Multiple codes are concatenated in alphabetical order. The default map expects axis-specific columns where the package provides them; run the relevant detector with by_axis = TRUE for spikes and interval jumps. The helper maps existing results and does not run any flag detector.

Usage

```
acti_flag_cdc_labels(
  data,
  flag_map = c(A = "flag_spike_X", B = "flag_spike_Y", C = "flag_spike_Z", D =
    "flag_device_max_X", E = "flag_device_max_Y", F = "flag_device_max_Z", G =
    "flag_device_min_X", H = "flag_device_min_Y", I = "flag_device_min_Z", J =
   "flag_contiguous_max_X", K = "flag_contiguous_max_Y", L = "flag_contiguous_max_Z", M
   = "flag_contiguous_min_X", N = "flag_contiguous_min_Y", O = "flag_contiguous_min_Z",
    P = "flag_impossible", Q = "flag_all_zero", R = "flag_repeated_samples", S =
    "flag_spike_second_X", T = "flag_spike_second_Y",
     U = "flag_spike_second_Z", V
   = "flag_adjacent_invalid", W = "flag_interval_jump_X", X = "flag_interval_jump_Y", Y
    = "flag_interval_jump_Z"),
  output = "PAXFLGSM",
  separator = ""
)
```

Arguments

data	Data containing logical flag columns.
flag_map	Named character vector mapping CDC letters to column names. Defaults to the package's axis-specific flag names and uses the mapped columns present in data, allowing a selected subset of detectors. A custom map must refer to existing columns.

output	Name of the output column.
separator	String inserted between multiple letters. Use "" for CDC-style compact values such as "ABP".

Value

Input data with a character column of CDC-style flag letters.

`acti_flag_contiguous_device_limit_runs`
Flag contiguous device-limit runs

Description

Marks axis-specific positive and negative device-limit samples in runs meeting `min_length`. Runs are broken at timestamp gaps based on the extracted or supplied sampling rate.

Usage

```
acti_flag_contiguous_device_limit_runs(
  data,
  dynamic_range = NULL,
  epsilon = 0.05,
  min_length = 1,
  sample_rate = NULL
)

acti_flag_contiguous_device_limits(
  data,
  dynamic_range = NULL,
  epsilon = 0.05,
  min_length = 1,
  sample_rate = NULL
)
```

Arguments

data	Raw accelerometer data.
dynamic_range	Device range; inferred when omitted.
epsilon	Margin inside the device limit, in g.
min_length	Minimum run length.
sample_rate	Sampling frequency; extracted from data when omitted.

Value

Input data with six axis- and sign-specific contiguous run columns.

acti_flag_device_limit_windows *Flag device-limit counts within time windows*

Description

Groups samples into fixed-duration windows from the first timestamp, counts values at the positive and negative device limits on each axis, and compares each count with `min_count`.

Usage

```
acti_flag_device_limit_windows(
  data,
  dynamic_range = NULL,
  epsilon = 0.05,
  min_count = 1,
  window_seconds = 60
)
```

Arguments

`data`	Raw accelerometer samples with a timestamp column.
`dynamic_range`	Device range; inferred when omitted.
`epsilon`	Margin inside the device limit, in g.
`min_count`	Minimum number of samples required within a window.
`window_seconds`	Window duration in seconds.

Value

One row per window with axis-specific flags.

acti_flag_qc *Flag Quality Control Values*

Description

`flag_qc_all()` is the pre-0.8.0 name of `acti_flag_qc_all()`, kept so that packages using it keep working. It will be deprecated in a future release; use `acti_flag_qc_all()` instead.

Usage

```
acti_flag_qc(
  data,
  dynamic_range = NULL,
  verbose = TRUE,
  flags = c("all", "spike", "interval_jump", "spike_second", "same_value",
    "device_limit", "all_zero", "impossible")
)

acti_flag_qc_all(
  data,
  dynamic_range = NULL,
  verbose = TRUE,
  flags = c("all", "spike", "interval_jump", "spike_second", "same_value",
    "device_limit", "all_zero", "impossible")
)

flag_qc_all(
  data,
  dynamic_range = NULL,
  verbose = TRUE,
  flags = c("all", "spike", "interval_jump", "spike_second", "same_value",
    "device_limit", "all_zero", "impossible")
)
```

Arguments

data	A data set of actigraphy
dynamic_range	dynamic range of the device, used to find the device limit.
verbose	print diagnostic messages
flags	the flags to run for QC. If you set this to "all", then all flags are run, as the default.

Value

A data set with a flags column (acti_flag_qc) or a number of columns starting with flag_* (acti_flag_qc_all)

Examples

```
res = acti_raw_data
out = acti_flag_qc(res)
```

`acti_flag_repeated_samples` *Flag repeated samples*

Description

Marks samples in runs of repeated three-axis observations. Options can exclude all-zero observations or require equal values across the three axes. Timestamp gaps break runs based on the supplied or extracted rate.

Usage

```
acti_flag_repeated_samples(
  data,
  sample_rate = NULL,
  min_length = 2,
  exclude_zero = FALSE,
  require_equal_axes = FALSE
)
```

Arguments

`data`	Raw accelerometer data.
`sample_rate`	Sampling frequency; extracted from data when omitted.
`min_length`	Minimum repeated run length in samples.
`exclude_zero`	Exclude samples where all axes are zero.
`require_equal_axes`	Require all axes to have equal values.

Value

Input data with `flag_repeated_samples`.

`acti_flag_spike` *Flag Spikes*

Description

Flag Spikes

Usage

```
acti_flag_spike(data, spike_size = 11, sample_rate = NULL, by_axis = FALSE)

acti_flag_interval_jump(
  data,
  verbose = TRUE,
  window_seconds = 1,
  min_repeats = 10,
  min_distinct = 3,
  min_separation = 0.5,
  by_axis = FALSE,
  digits = 3,
  chunk_windows = 50000L
)

acti_flag_spike_second(
  data,
  spike_size = 11,
  window_seconds = 1,
  min_spikes = 13,
  max_change_seconds = 0.1,
  sample_rate = NULL,
  by_axis = FALSE
)

acti_flag_device_limit(data, dynamic_range = NULL, epsilon = 0.05)

acti_flag_adjacent_device_limit_samples(
  data,
  dynamic_range = NULL,
  epsilon = 0.05,
  sample_rate = NULL
)

acti_flag_contiguous_device_limit(
  data,
  dynamic_range = NULL,
  epsilon = 0.05,
  sample_rate = NULL
)

acti_flag_same_value(data, min_length = 1, sample_rate = NULL)

acti_flag_all_zero(data, min_length = 3, sample_rate = NULL)

acti_flag_impossible(
  data,
  min_length = 7,
```

```
  change_threshold = 0.01,
  magnitude_threshold = 1.25,
  sample_rate = NULL,
  dynamic_range = NULL
)
```

Arguments

data	A data set of actigraphy
spike_size	size of "spike" - which is the absolute difference in contiguous observations on a single axis
sample_rate	Sampling frequency; extracted from data when omitted.
by_axis	If TRUE, also return one indicator per axis.
verbose	print diagnostic messages
window_seconds	Width of each window used by acti_flag_spike_second.
min_repeats	Minimum occurrences of an axis value in a window.
min_distinct	Minimum number of frequent axis values required.
min_separation	Minimum separation between retained values.
digits	Number of decimal places used when grouping axis values.
chunk_windows	Number of windows processed together to bound memory.
min_spikes	Minimum number of qualifying extrema in a window.
max_change_seconds	Maximum time between an extremum and a qualifying large change; the CDC rule uses less than 0.1 seconds.
dynamic_range	Device dynamic range. When omitted, it is read from the data with get_dynamic_range(); data-based estimates are rejected.
epsilon	A small adjustment so that if values are within the device limit, but minus epsilon, still flagged as hitting the limit. For example, if dynamic_range = c(-6, 6) and epsilon = 0.05, then any value <= -5.95 or >= 5.95 gravity units will be flagged
min_length	minimum length of the condition for contiguous samples. If min_length = 7, then at least 7 TRUEs in a row is required, any stretches of single TRUE values or 2 TRUE followed by FALSE, will be set to FALSE.
change_threshold	Maximum absolute change between adjacent samples on each axis for a stable run to qualify. The default is 0.01 g.
magnitude_threshold	Minimum vector magnitude required at both ends of each stable adjacent pair. The default is 1.25 g.

Details

These functions provide configurable sample-level and window-level signal indicators. Thresholds and run or window lengths are arguments; they do not encode a specific device or study's quality-control scheme. When continuity is checked, sample_rate is extracted with get_sample_rate() unless supplied explicitly.

Value

A data set back

Note

`acti_flag_spike` compares adjacent observations by axis. The `acti_flag_spike_second` checks for repeated local extrema followed by a large change within the requested time, then counts them within each window.

Examples

```
res = acti_raw_data
res = acti_flag_spike(res)
res = acti_flag_interval_jump(res)
res = acti_flag_spike_second(res)
res = acti_flag_same_value(res)
res = acti_flag_device_limit(res)
res = acti_flag_all_zero(res)
res = acti_flag_impossible(res)
```

`acti_hms_to_minute` *Convert an HMS Time to a Minute Index*

Description

Convert an HMS Time to a Minute Index

Usage

```
acti_hms_to_minute(x, start = 1L)
```

Arguments

`x`	An `hms`, `POSIXt`, or `HH:MM[:SS]` character vector.
`start`	The index assigned to midnight, either `0` or `1`.

Value

An integer minute-of-day index.

Note

This is a strict `hms` entry point for callers that have already separated time of day from date and want the standard activity-day index.

Examples

```
acti_hms_to_minute(hms::hms(hours = c(0, 23), minutes = c(0, 59)))
```

acti_minute_to_hms *Convert a Minute Index to an HMS Time*

Description

Convert a Minute Index to an HMS Time

Usage

```
acti_minute_to_hms(minute, start = 1L)
```

Arguments

minute	Whole-minute positions in a day.
start	The index assigned to midnight, either 0 or 1.

Value

An hms vector.

Note

This is the inverse of acti_time_to_minute() for readable axis labels and for joining minute-indexed results back to a time-of-day value.

Examples

```
acti_minute_to_hms(c(1, 721, 1440))
acti_minute_to_hms(c(0, 1439), start = 0)
```

acti_raw_data *Example Actigraphy/Activity Raw Data*

Description

Example Actigraphy/Activity Raw Data

Usage

```
acti_raw_data
```

Format

A data.frame with the columns

time time column

X X-axis column

Y Y-axis column

Z Z-axis column

`acti_repeat_time_of_day` *Repeat a Time of Day Across a Recording*

Description

Repeat a Time of Day Across a Recording

Usage

```
acti_repeat_time_of_day(seconds, time, timezone = NULL)
```

Arguments

`seconds`	Seconds after midnight, or an `hms`/`HH:MM[:SS]` time.
`time`	A `POSIXt` recording time vector used to determine dates and timezone.
`timezone`	Optional Olson timezone; by default the timezone of `time`.

Value

A `POSIXct` vector with one occurrence per recording date.

Note

This supplies daily plotting boundaries such as a fixed bedtime or wake time for every calendar date present in a recording.

Examples

```
time = as.POSIXct(c("2020-01-01 12:00:00", "2020-01-03 12:00:00"), tz = "UTC")
acti_repeat_time_of_day("01:30", time)
```

`acti_resample` *Resample 3-axial input signal to a specific sampling rate*

Description

Resample 3-axial input signal to a specific sampling rate

Usage

```
acti_resample(data, sample_rate, method = "linear", ...)

acti_resample_to_time(data, times, method = "linear", ...)
```

Arguments

`data`	A `data.frame` with a column for time in `POSIXct` (usually `time`), and `X`, `Y`, `Z`
`sample_rate`	sampling frequency, coercible to an integer. This is the sampling rate you're sampling the data *into*.
`method`	method for interpolation. Options are `"linear"/"constant"`, which uses `stats::approx`, or one of `"fmm"`, `"periodic"`, `"natural"`, `"monoH.FC"`, `"hyman"`, which uses `stats::spline`
`...`	additional arguments to pass to `stats::approx()` or stats::spline
`times`	a vector of `POSIXct` date/time values to interpolate the data to

Value

A `data.frame`/`tibble` of `time` and `X`, `Y`, `Z`.

Examples

```
old = options(digits.secs = 3)
x = acti_raw_data
res = acti_resample(data = x, sample_rate = 80)
res = acti_resample(data = x, sample_rate = 100)
res = acti_resample(data = x, sample_rate = 1)
res = acti_resample_to_time(
  data = x,
  times = lubridate::floor_date(x$time, unit = "1 sec"),
)
res_nat = acti_resample_to_time(
  data = x,
  times = lubridate::floor_date(x$time, unit = "1 sec"),
  method = "natural"
)
options(old)
```

acti_separate_time *Separate Times into Date, Hour, and Minute*

Description

Separate Times into Date, Hour, and Minute

Usage

```
acti_separate_time(data)

acti_separate_times(data)

acti_create_date(data)

acti_create_hour(data)

acti_create_minute(data)
```

Arguments

data	a data.frame with a time column

Value

A data.frame with date, hour, minute, and day columns

Examples

```
library(actibase)
acti_separate_time(acti_raw_data)
acti_create_date(acti_raw_data)
acti_create_hour(acti_raw_data)
acti_create_minute(acti_raw_data)
```

acti_split_intervals_at_midnight *Split Intervals at Midnight*

Description

Split Intervals at Midnight

Usage

```
acti_split_intervals_at_midnight(start, end, timezone = NULL)
```

Arguments

start, end	Equally sized POSIXt vectors.
timezone	Optional Olson timezone used to define midnight.

Value

A data frame with input interval number and split start/end.

Note

Sleep, wear, and activity episodes can span dates; splitting them makes their durations and faceted daily displays unambiguous.

Examples

```
start = as.POSIXct("2020-01-01 23:30:00", tz = "UTC")
end = as.POSIXct("2020-01-02 00:30:00", tz = "UTC")
acti_split_intervals_at_midnight(start, end)
```

---

acti_standardize_data *Standardize the Accelerometry Data*

Description

Standardize the Accelerometry Data

Usage

```
acti_standardize_data(
  data,
  subset_xyz = TRUE,
  colname_time = "time",
  lower_case = FALSE,
  check_xyz = TRUE
)

acti_standardise_data(
  data,
  subset_xyz = TRUE,
  colname_time = "time",
  lower_case = FALSE,
  check_xyz = TRUE
)
```

Arguments

data	A data.frame with an optional time column and three accelerometer axes. Axes can be named X/Y/Z, x/y/z, acc_X/acc_Y/acc_Z, acc_x/acc_y/acc_z, or acceleration_X/acceleration_Y/acceleration_Z (with lowercase axis letters also accepted).
subset_xyz	should only the time (if available) and XYZ be subset?
colname_time	column name of header for time
lower_case	flag to make sure all columns are lower case.
check_xyz	Check if X/Y/Z is in the data

Details

When a complete canonical triplet and a complete prefixed triplet (acc_* or acceleration_*) are both present, the canonical columns are used and a single warning is issued. A data frame containing both X and x (or the analogous pair for Y or Z) is rejected as ambiguous.

Value

A data.frame with X/Y/Z and a time in time (if available).

Examples

```
acti_standardize_data(acti_raw_data)
acti_standardise_data(acti_raw_data)
```

---

acti_standardize_epoch

Standardize an Epoch Length

Description

Converts an epoch given as a number, a string such as "1 min", "30s", "5 seconds", or "1 hour", a lubridate Duration or Period, or a difftime into a consistent representation: either a number in a given unit (e.g. 60 seconds) or a string that can be passed to lubridate::floor_date() (e.g. "1 min").

Usage

```
acti_standardize_epoch(
  epoch,
  unit = c("seconds", "minutes", "hours", "days", "auto"),
  output = c("numeric", "character"),
  input_unit = c("seconds", "minutes", "hours", "days")
)
```

```
acti_standardise_epoch(
  epoch,
  unit = c("seconds", "minutes", "hours", "days", "auto"),
  output = c("numeric", "character"),
  input_unit = c("seconds", "minutes", "hours", "days")
)
```

Arguments

epoch	A single epoch length: a positive number (in input_unit), a string parseable by lubridate::duration() ("1 min", "1min", "30s", "2 hours"; "hr" and the lubridate arbitrary units such as "90 aseconds" are also accepted), a Duration, a Period, or a difftime. Calendar units without a fixed length (months, quarters, years, ...) are not epochs and give an error.
unit	Unit to standardize the epoch to: one of "seconds", "minutes", "hours", "days", or "auto". "auto" is only available for output = "character" and picks the largest unit that lubridate::floor_date() can round with (e.g. "1 min" rather than "60 secs"), falling back to "aseconds" for lengths such as 90 seconds. Defaults to "seconds" for numeric output and "auto" for character output. Singular and abbreviated unit names are matched to the unit they name: "second", "sec", and "secs" are treated as "seconds"; "minute", "min", and "mins" as "minutes"; "hour", "hr", and "hrs" as "hours"; and "day" as "days". Other names, including partial ones such as "m", are an error.
output	Type of output: "numeric" returns the length in unit, "character" returns a string such as "30 secs", "1 min", or "2 hours", which lubridate::floor_date(), seq(), cut(), and MIMSunit all understand.
input_unit	Unit of epoch when epoch is numeric (or a string with no unit, such as "60"). Ignored otherwise. Abbreviations are matched as for unit.

Value

A numeric value or a single string.

Note

The "aseconds" fallback for output = "character" relies on the absolute-unit rounding that lubridate::floor_date() has supported since lubridate 1.9.0.

Examples

```
acti_standardize_epoch(60)
acti_standardize_epoch("1 min")
acti_standardize_epoch("1 min", unit = "minutes")
acti_standardize_epoch(30, output = "character")
acti_standardize_epoch(60, output = "character")
acti_standardize_epoch(5, input_unit = "minutes")
acti_standardize_epoch("2 hours", unit = "minutes", output = "character")
acti_standardize_epoch(lubridate::minutes(5))
acti_standardize_epoch(as.difftime(30, units = "secs"), output = "character")
```

acti_tidy_axes *Tidy axes to a long format*

Description

Tidy axes to a long format

Usage

```
acti_tidy_axes(data, colname_time = "time")
```

Arguments

data	An object with columns a time column X, Y, and Z or an object of class AccData
colname_time	column name of header for time

Value

A long data set with time, axis, and value

Examples

```
long = acti_tidy_axes(acti_raw_data)
```

acti_time_of_day *Extract the Time of Day*

Description

Extract the Time of Day

Usage

```
acti_time_of_day(time, timezone = NULL)
```

Arguments

time	A POSIXt vector.
timezone	Optional Olson timezone in which to extract time of day.

Value

An hms vector.

Note

This explicitly applies the recording or requested timezone before dropping the date, which prevents a timestamp from being assigned to the wrong activity-day minute.

Examples

```
time = as.POSIXct("2020-01-01 05:30:00", tz = "UTC")
acti_time_of_day(time, timezone = "America/New_York")
```

---

acti_time_to_minute *Convert a Time of Day to a Minute Index*

Description

Convert a Time of Day to a Minute Index

Usage

```
acti_time_to_minute(x, start = 1L, timezone = NULL)
```

Arguments

x	An hms, POSIXt, or HH:MM[:SS] character vector.
start	The index assigned to midnight, either 0 or 1.
timezone	Optional Olson timezone for POSIXt values.

Value

An integer minute-of-day index.

Note

This converts each minute of a day to either 0 through 1439 or 1 through 1440, the two common minute-of-day conventions.

Examples

```
acti_time_to_minute(c("00:00", "12:00", "23:59"))
acti_time_to_minute("00:00", start = 0)
```

as_convert_safe *Convert vectors ensuring no new NA*

Description

Convert vectors ensuring no new NA

Usage

```
as_convert_safe(x, ..., func = lubridate::as_datetime)

as_date_safe(x, ...)

as_datetime_safe(x, ...)
```

Arguments

x	a vector
...	additional arguments to pass to func
func	the function to use to transform the vector x

Value

A converted vector the same length as x or errors if there are introduced NAs.

create_day_inclusion *Create Day-Level Inclusion information*

Description

Create Day-Level Inclusion information

Usage

```
create_day_inclusion(data, min_required = 1368L)

add_day_inclusion(data, ...)
```

Arguments

data	A data.frame with the columns time
min_required	Number of minutes required in a day to be called included
...	arguments to pass to create_day_inclusion when using add_day_inclusion

Value

A data.frame for each day with information of number of minutes observed and included

Examples

```
data = acti_raw_data |>
  dplyr::mutate(r = sqrt(X^2 + Y^2 + Z^2),
                time = lubridate::floor_date(time, "1 minute")) |>
  dplyr::group_by(time) |>
  dplyr::summarise(r = sum(r), .groups = "drop") |>
  dplyr::mutate(wear = r > 4000)

res = create_day_inclusion(data)
```

---

get_dynamic_range *Get Dynamic Range*

Description

Get Dynamic Range

Usage

```
get_dynamic_range(data, dynamic_range = NULL, flag_estimated = FALSE)
```

Arguments

data	An AccData object from an actigraphy reader
dynamic_range	the dynamic range. If this is not NULL, then it will be guess from the header or the data
flag_estimated	if TRUE, then the output will have the attribute "estimated", which is a logical indicated if it was found or estimated

Value

A length-2 numeric vector, or the original dynamic range (no checking done)

get_sample_rate *Get Sample Rate*

Description

Get Sample Rate

Usage

```
get_sample_rate(data, sample_rate = NULL, flag_estimated = FALSE)
```

Arguments

data	A data set of actigraphy/activity data
sample_rate	the sample rate. If this is not NULL, then it will be guess from the header or the data or the data separation
flag_estimated	if TRUE, then the output will have the attribute "estimated", which is a logical indicated if it was found or estimated

Value

A length-1 numeric vector

get_transformations *Get Transformations*

Description

Get Transformations

Usage

```
get_transformations(data)

prefix_transformations(transformations, prefix = NULL)

set_transformations(data, transformations, add = TRUE, prefix = NULL)
```

Arguments

data	data set of data, usually time and X/Y/Z.
transformations	character string of transformations
prefix	if not NULL, the prefix plus : would be pasted to the transformations.
add	Add the transformations to those already there in data

Value

set_transformations returns the data, with the transformations attribute updated and set_transformations returns the attribute transformations

---

is.AccData *Is the object of class* AccData

Description

Is the object of class AccData

Usage

```
is.AccData(x)
```

Arguments

x	object to test

Value

Logical(1)

Examples

```
is.AccData(mtcars)
```

---

mark_condition *Mark a Condition of a Specified Minimum Length*

Description

Mark a Condition of a Specified Minimum Length

Usage

```
mark_condition(x, min_length = 1)
```

Arguments

x	A logical vector
min_length	minimum length, contiguous TRUE values required

Value

A logical vector

Examples

```
x = c(FALSE, TRUE, TRUE, FALSE, FALSE, rep(TRUE, 10), FALSE, rep(TRUE, 20))
mark_condition(x)
mark_condition(x, 2)
mark_condition(x, 5)
mark_condition(x, 15)
```

---

strip_hour_shift *Strip Hour Shift from Character Time Vector*

Description

Strip Hour Shift from Character Time Vector

Usage

```
strip_hour_shift(x, max_index = 2L)
```

Arguments

x	character vector with times that may include hour shifts (e.g., "2019-01-01 12:00+03:00")
max_index	maximum index to grab the shift from after splitting on spaces, default is 2 (e.g., "2019-01-01 12:00")

Value

A character vector with the +/- hour shift removed

Examples

```
strip_hour_shift(c("2019-01-01 12:00+03:00", "2019-01-01 12:00-04:00"))
```

---

xyz *Vector of X, Y, Z, and maybe time*

Description

Vector of X, Y, Z, and maybe time

Usage

```
xyz

xyzt

txyz
```

Format

A vector with 3 or 4 elements, which are:

X value to extract X column

Y value to extract Y column

Z value to extract Y column

time value to extract time column

An object of class `character` of length 4.

An object of class `character` of length 4.

Index

∗ **datasets**
acti_raw_data, 14
xyz, 27

acti_create_date *(*acti_separate_time*)*, 17
acti_create_hour *(*acti_separate_time*)*, 17
acti_create_minute *(*acti_separate_time*)*, 17
acti_day_index, 2
acti_day_start, 3
acti_fill_zeroes *(*acti_fill_zeros*)*, 4
acti_fill_zeros, 4
acti_flag_adjacent_device_limit_samples *(*acti_flag_spike*)*, 10
acti_flag_adjacent_invalid_periods, 5
acti_flag_all_zero *(*acti_flag_spike*)*, 10
acti_flag_cdc_labels, 6
acti_flag_contiguous_device_limit *(*acti_flag_spike*)*, 10
acti_flag_contiguous_device_limit_runs, 7
acti_flag_contiguous_device_limits *(*acti_flag_contiguous_device_limit_runs*)*, 7
acti_flag_device_limit *(*acti_flag_spike*)*, 10
acti_flag_device_limit_windows, 8
acti_flag_impossible *(*acti_flag_spike*)*, 10
acti_flag_interval_jump *(*acti_flag_spike*)*, 10
acti_flag_qc, 8
acti_flag_qc_all *(*acti_flag_qc*)*, 8
acti_flag_repeated_samples, 10
acti_flag_same_value *(*acti_flag_spike*)*, 10
acti_flag_spike, 10
acti_flag_spike_second *(*acti_flag_spike*)*, 10
acti_hms_to_minute, 13
acti_minute_to_hms, 14
acti_raw_data, 14
acti_remove_leading_zeroes *(*acti_fill_zeros*)*, 4
acti_remove_leading_zeros *(*acti_fill_zeros*)*, 4
acti_repeat_time_of_day, 15
acti_resample, 16
acti_resample_to_time *(*acti_resample*)*, 16
acti_separate_time, 17
acti_separate_times *(*acti_separate_time*)*, 17
acti_split_intervals_at_midnight, 17
acti_standardise_data *(*acti_standardize_data*)*, 18
acti_standardise_epoch *(*acti_standardize_epoch*)*, 19
acti_standardize_data, 18
acti_standardize_epoch, 19
acti_tidy_axes, 21
acti_time_of_day, 21
acti_time_to_minute, 22
acti_time_to_minute(), *14*
add_day_inclusion, *23*
add_day_inclusion *(*create_day_inclusion*)*, 23
as_convert_safe, 23
as_date_safe *(*as_convert_safe*)*, 23
as_datetime_safe *(*as_convert_safe*)*, 23

create_day_inclusion, *23*, 23
cut(), *20*

flag_qc_all *(*acti_flag_qc*)*, 8

get_dynamic_range, 24

get_sample_rate, 25
get_sample_rate(), *5*
get_transformations, 25

is.AccData, 26

lubridate::duration(), *20*
lubridate::floor_date(), *3*, *19*, *20*

mark_condition, 26

prefix_transformations
        *(*get_transformations*)*, 25

seq(), *20*
set_transformations, *26*
set_transformations
        *(*get_transformations*)*, 25
stats::approx(), *16*
stats::spline, *16*
strip_hour_shift, 27

txyz *(*xyz*)*, 27

xyz, 27
xyzt *(*xyz*)*, 27