Package: actisomni (via r-universe)

October 16, 2026

Type Package

Title Create Sleep Measures of Actigraphy and Activity Analysis

Version 0.2.0

Description Estimates sleep from wrist accelerometry and epoch-level activity data. Implements sleep-period-time guiders based on angle changes, posture, least-active periods, sustained inactivity, low activity, and diary or fixed time windows; combines guider windows with sustained-inactivity bouts to produce sleep/wake labels and consensus estimates. Provides diary helpers and Tudor-Locke sleep-period summaries, and interfaces to the 'asleep' and 'sleeper' machine-learning estimators. The angle-based and sustained-inactivity methods are described by van Hees et al. (2015) <doi:10.1371/journal.pone.0142533> and van Hees et al. (2018) <doi:10.1038/s41598-018-31266-z>; the sleep-period summary method by Tudor-Locke et al. (2014) <doi:10.1139/apnm-2013-0173>; and the model estimators by Yuan et al. (2024) <doi:10.1038/s41746-024-01065-0> and Sundararajan et al. (2021) <doi:10.1038/s41598-020-79217-x>.

License GPL-3

Depends R (>= 4.1.0)

Suggests testthat, utils, covr, knitr, httr, rmarkdown, data.table, reticulate, callr, GGIR, rlang, actimetrics

Encoding UTF-8

Imports actibase, actiread, dplyr, lubridate, assertthat, tibble, stats, asleep, sleeper, actigraph.sleepr

Config/roxygen2/version 8.0.0

URL https://jhuwit.github.io/actisomni/

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

NeedsCompilation no

Author John Muschelli [aut, cre]

Maintainer John Muschelli <muschellij2@gmail.com>

Repository https://cran.r-universe.dev

Date/Publication 2026-10-16 12:30:07 UTC

RemoteUrl https://github.com/cran/actisomni

RemoteRef HEAD

RemoteSha b95e459a70eafd5f77fba47c58ad32e4758854a8

Contents

acti_asleep 2
acti_sleep_diary 4
acti_sleep_ensemble 4
acti_sleep_fuse_labels 5
acti_sleep_guider 6
acti_sleep_guider_requirements 7
acti_sleep_hdcza 8
acti_sleep_hlrb 9
acti_sleep_horangle 9
acti_sleep_l5 10
acti_sleep_label 11
acti_sleep_labels 12
acti_sleep_notworn 13
acti_sleep_setwindow 13
acti_sleep_sib 14
acti_sleep_tudor_locke 15
acti_sleep_tudor_locke_diary 16
acti_sleeper 17
py_acti_asleep 17
py_acti_sleeper 18

Index **20**

`acti_asleep` *Estimate sleep with the asleep model*

Description

Runs `asleep::asleep()` on complete, time-ordered raw triaxial acceleration and converts its predictions to the common `acti_sleep_estimate` format.

Usage

```
acti_asleep(
  data,
  min_wear_hours = 22L,
  time_shift = "0",
  report_light_and_temp = FALSE,
  pytorch_device = c("cpu", "cuda:0"),
  sample_rate = NULL,
  verbose = TRUE,
  force_download = FALSE
)

acti_calculate_asleep(
  data,
  min_wear_hours = 22L,
  time_shift = "0",
  report_light_and_temp = FALSE,
  pytorch_device = c("cpu", "cuda:0"),
  sample_rate = NULL,
  verbose = TRUE,
  force_download = FALSE
)
```

Arguments

`data`	A data frame containing timestamps and triaxial acceleration, or a readable accelerometry file path. File paths are passed unchanged to `asleep::asleep()`.
`min_wear_hours`	Minimum daily wear time used by `asleep` summaries.
`time_shift`	Clock-time correction passed to `asleep::asleep()`.
`report_light_and_temp`	Whether to request light and temperature output.
`pytorch_device`	Device used for prediction.
`sample_rate`	Optional input sample rate.
`verbose`	Whether `asleep` prints progress messages.
`force_download`	Whether to re-download `asleep` model files.

Value

An `acti_sleep_estimate` tibble with `time`, `sleep`, `sleep_probability`, `sleep_stage`, `nonwear`, and `method`. The asleep backend currently does not return probabilities, so `sleep_probability` is `NA`.

acti_sleep_diary *Identify a diary-defined sleep window*

Description

Identify a diary-defined sleep window

Usage

```
acti_sleep_diary(data, onset, wakeup)
```

Arguments

data	A data frame containing timestamps.
onset, wakeup	POSIXct values defining a diary sleep window.

Value

An acti_sleep_guider object. Missing diary times return an empty window rather than imputing either boundary.

Examples

```
data <- actimetrics::acti_count_data
diary_window <- acti_sleep_diary(
  data, onset = data$time[5], wakeup = data$time[30]
)
```

acti_sleep_ensemble *Run a sleep-label ensemble from raw or epoch data*

Description

This convenience workflow detects or accepts sustained inactivity bouts, runs compatible sleep guiders, creates one sleep label per guider, then produces a consensus. Raw triaxial data use HDCZA, HorAngle, and a fixed 22:00–08:00 window by default; L5 is included only when the data span its requested duration. Epoch data use NotWorn and the fixed window, adding L5 when possible. Set guiders to choose a subset explicitly.

Usage

```
acti_sleep_ensemble(
  data,
  guiders = NULL,
  epoch = "5 seconds",
  sib_method = "vanHees2015",
  fusion = c("staple", "majority"),
  sleepwindow_type = c("SPT", "TimeInBed"),
  ...
)
```

Arguments

`data`	Raw triaxial or regular epoch-level data. Epoch data should have a `sib` column or a conventional activity column.
`guiders`	Guider names. Defaults depend on whether raw acceleration is available.
`epoch`	Raw-data aggregation epoch.
`sib_method`	SIB method, passed to `acti_sleep_sib()`.
`fusion`	Consensus rule: `"staple"` or `"majority"`.
`sleepwindow_type`	SPT or TimeInBed overlap rule.
`...`	Passed to the individual guider functions.

Value

A list of time-indexed data. `epoch_data` includes the van Hees `sib`; each guider has a `label_data` element with `time` and `window`; `labels` has `time` plus one logical `sleep_<guider>` column per guider; `fused` has `time`, probability, and consensus label; and `label_data` combines all these analysis columns.

Examples

```
raw <- actiread::acti_read_gt3x(actiread::acti_example_gt3x(), verbose = FALSE)
result <- acti_sleep_ensemble(raw)
```

---

acti_sleep_fuse_labels

Fuse multiple sleep/wake labels with STAPLE

Description

STAPLE estimates a latent consensus sleep label and each input labeler's sensitivity and specificity using expectation-maximisation.

Usage

```
acti_sleep_fuse_labels(
  labels,
  threshold = 0.5,
  tolerance = 1e-06,
  max_iterations = 200L
)
```

Arguments

`labels`	A data frame, matrix, or tibble of logical/binary sleep-label columns, such as the output of `acti_sleep_labels()`.
`threshold`	Posterior sleep-probability threshold for the fused label.
`tolerance`, `max_iterations`	STAPLE convergence controls.

Value

A tibble with `sleep_probability` and binary `sleep` columns. If `labels` includes `time` or `epoch`, that index is retained. The fitted STAPLE parameters are attached as the `staple` attribute.

Examples

```
labels <- data.frame(
  sleep_l5 = c(FALSE, TRUE, TRUE, FALSE),
  sleep_fixed = c(FALSE, TRUE, FALSE, FALSE)
)
acti_sleep_fuse_labels(labels)
```

`acti_sleep_guider` *Dispatch a sleep-period-time guider from data*

Description

For `"HDCZA"` and `"HorAngle"`, supply raw triaxial acceleration; the selected function derives a regular epoch-level posture angle. The remaining methods use regular epoch-level activity, SIB, or timestamp data. Use `acti_sleep_guider_requirements()` for a method-by-method summary.

Usage

```
acti_sleep_guider(
  data,
 method = c("HDCZA", "HorAngle", "L5", "setwindow", "HLRB", "NotWorn", "sleeplog"),
  ...
)
```

Arguments

`data`	A raw or epoch-level data frame, according to `method`.
`method`	One of `"HDCZA"`, `"HorAngle"`, `"L5"`, `"setwindow"`, `"HLRB"`, `"NotWorn"`, or `"sleeplog"`.
`...`	Method-specific arguments, such as `threshold`, `start_hour`, `onset`, and `wakeup`.

Value

An `acti_sleep_guider` object.

Examples

```
data <- actimetrics::acti_count_data
data <- data[rep(seq_len(nrow(data)), length.out = 24 * 60), ]
data$time <- data$time[1] + (seq_len(nrow(data)) - 1) * 60
guider <- acti_sleep_guider(data, method = "L5")
```

---

acti_sleep_guider_requirements

Describe data requirements for sleep guiders

Description

HDCZA and HorAngle take raw triaxial acceleration and create their own regular epoch-level angle. All other guiders take a regularly spaced, epoch-level data frame and do not require raw acceleration data.

Usage

```
acti_sleep_guider_requirements()
```

Value

A tibble describing the required data for each guider.

Examples

```
acti_sleep_guider_requirements()
```

acti_sleep_hdcza *Identify a sleep window with HDCZA*

Description

Implements the HDCZA sleep-period-time guider for wrist accelerometry. It finds prolonged periods of little change in the supplied z-axis angle, joins interruptions shorter than an hour, and returns the longest resulting block. The default threshold is the 0.2 degrees specified in the GGIR guider documentation.

Usage

```
acti_sleep_hdcza(
  data,
  epoch = "5 seconds",
  threshold = 0.2,
  ignore_invalid = FALSE,
  min_block_minutes = 30,
  max_gap_minutes = 60
)
```

Arguments

`data`	A data frame of raw accelerometer observations.
`epoch`	Epoch used to average raw angles, passed to `lubridate::floor_date()`.
`threshold`	Maximum five-minute rolling median absolute angle change.
`ignore_invalid`	Whether invalid observations are treated as movement; use `NA` to treat them as no movement.
`min_block_minutes`	Minimum low-movement block duration.
`max_gap_minutes`	Maximum interruption bridged between blocks.

Value

An `acti_sleep_guider` object. `window` is the selected SPT guider; `crude_window` retains all candidate blocks before choosing the longest.

Examples

```
# The bundled actiread recording contains raw X, Y, and Z acceleration.
raw <- actiread::acti_read_gt3x(actiread::acti_example_gt3x(), verbose = FALSE)
hdcza <- acti_sleep_hdcza(raw, epoch = "5 seconds")
```

acti_sleep_hlrb *Identify the longest rest bout from sustained inactivity bouts*

Description

Identify the longest rest bout from sustained inactivity bouts

Usage

```
acti_sleep_hlrb(data)
```

Arguments

data	A data frame of regular epochs.

Value

An `acti_sleep_guider` object. `window` marks the longest selected rest bout, `crude_window` marks all bouts before selection, and `smoothed_sib` is the logical SIB classification after two-hour smoothing.

Examples

```
data <- actimetrics::acti_count_data
data <- data[rep(seq_len(nrow(data)), length.out = 24 * 60), ]
data$time <- data$time[1] + (seq_len(nrow(data)) - 1) * 60
# In practice, use a validated SIB classification rather than this example.
sib <- data$counts == 0
data$sib <- data$counts == 0
longest_rest <- acti_sleep_hlrb(data)
```

acti_sleep_horangle *Identify a sleep window from horizontal posture*

Description

Identify a sleep window from horizontal posture

Usage

```
acti_sleep_horangle(
  data,
  epoch = "5 seconds",
  longitudinal_axis = "Z",
  horizontal_threshold = 45,
  ignore_invalid = FALSE,
  min_block_minutes = 30,
  max_gap_minutes = 60
)
```

Arguments

`data` A data frame of raw accelerometer observations.

`epoch` Epoch used to average raw angles, passed to `lubridate::floor_date()`.

`longitudinal_axis` Raw axis used as the longitudinal axis: `"X"`, `"Y"`, or `"Z"`.

`horizontal_threshold` Absolute-angle limit defining horizontal posture.

`ignore_invalid` Whether invalid observations are treated as movement; use `NA` to treat them as no movement.

`min_block_minutes` Minimum low-movement block duration.

`max_gap_minutes` Maximum interruption bridged between blocks.

Value

An `acti_sleep_guider` object. `window` marks the longest selected horizontal-posture block, and `crude_window` marks all candidate blocks. `horizontal` is the epoch-level posture flag, `threshold` is the angle limit, and `epoch_data` contains the averaged angle and invalid flag.

Examples

```
raw <- actiread::acti_read_gt3x(actiread::acti_example_gt3x(), verbose = FALSE)
horizontal <- acti_sleep_horangle(raw, epoch = "5 seconds")
```

---

acti_sleep_l5 *Identify a sleep window from the least-active five hours*

Description

Identify a sleep window from the least-active five hours

Usage

```
acti_sleep_l5(data, l5_hours = 5, window_hours = 12)
```

Arguments

data	A data frame of regular activity epochs.
l5_hours	Duration of the least-active period.
window_hours	Duration of the guider centred on L5.

Value

An acti_sleep_guider object. window marks the window centred on the least-active period; l5_start_index and l5_activity_sum identify that period's starting epoch and total activity.

Examples

```
# Repeat the bundled minute-level count data to form a complete example day.
data <- actimetrics::acti_count_data
data <- data[rep(seq_len(nrow(data)), length.out = 24 * 60), ]
data$time <- data$time[1] + (seq_len(nrow(data)) - 1) * 60
l5 <- acti_sleep_l5(data)
```

---

acti_sleep_label *Convert a sleep guider to epoch-level sleep/wake labels*

Description

A guider defines the likely main sleep window; it is not a sleep/wake classifier itself. This function applies GGIR's final overlap step to a sustained-inactivity-bout (sib) classification. With the default "SPT" rule, an entire SIB is sleep when any part overlaps the guider. With "TimeInBed", an SIB must be wholly inside the guider.

Usage

```
acti_sleep_label(sib, guider, sleepwindow_type = c("SPT", "TimeInBed"))
```

Arguments

sib	Logical or 0/1 sustained-inactivity-bout classification at the same regular epochs as guider$window.
guider	An acti_sleep_guider object.
sleepwindow_type	Whether the guider approximates sleep-period time ("SPT") or time in bed ("TimeInBed").

Value

A logical vector: TRUE for sleep and FALSE for wake.

Examples

```
data <- actimetrics::acti_count_data
data <- data[rep(seq_len(nrow(data)), length.out = 24 * 60), ]
data$time <- data$time[1] + (seq_len(nrow(data)) - 1) * 60
guider <- acti_sleep_l5(data)
sib <- data$counts == 0
sleep <- acti_sleep_label(sib, guider)
```

---

acti_sleep_labels *Create sleep/wake labels for multiple guiders*

Description

Create sleep/wake labels for multiple guiders

Usage

```
acti_sleep_labels(sib, guiders, sleepwindow_type = c("SPT", "TimeInBed"))
```

Arguments

`sib`	Logical or 0/1 sustained-inactivity-bout classification.
`guiders`	A named list of `acti_sleep_guider` objects. All guider windows must use the same epoch grid as `sib`.
`sleepwindow_type`	The SPT or TimeInBed overlap rule.

Value

A tibble with one logical `sleep_<guider>` column per guider.

Examples

```
data <- actimetrics::acti_count_data
data <- data[rep(seq_len(nrow(data)), length.out = 24 * 60), ]
data$time <- data$time[1] + (seq_len(nrow(data)) - 1) * 60
guiders <- list(L5 = acti_sleep_l5(data), fixed = acti_sleep_setwindow(data))
labels <- acti_sleep_labels(data$counts == 0, guiders)
```

acti_sleep_notworn *Identify a low-activity window for no-night-wear protocols*

Description

Identify a low-activity window for no-night-wear protocols

Usage

```
acti_sleep_notworn(data, min_block_minutes = 30, max_gap_minutes = 60)
```

Arguments

`data`	A data frame of regular activity epochs.
`min_block_minutes`	Minimum duration of a low-activity block.
`max_gap_minutes`	Maximum interruption bridged between blocks.

Value

An `acti_sleep_guider` object. `window` marks the longest selected low-activity block, and `crude_window` marks all candidate blocks. `smoothed_activity` contains five-minute rolling activity values, `low_activity` flags candidate epochs, and `threshold` is the activity cutoff used to identify them.

Examples

```
data <- actimetrics::acti_count_data
data <- data[rep(seq_len(nrow(data)), length.out = 24 * 60), ]
data$time <- data$time[1] + (seq_len(nrow(data)) - 1) * 60
# `counts` is an epoch-level activity metric.
no_night_wear <- acti_sleep_notworn(data)
```

acti_sleep_setwindow *Identify a fixed daily sleep window*

Description

Identify a fixed daily sleep window

Usage

```
acti_sleep_setwindow(data, start_hour = 22, end_hour = 8)
```

Arguments

`data` A data frame containing timestamps.

`start_hour`, `end_hour` Start and end clock hours (0–24).

Value

An `acti_sleep_guider` object.

Examples

```
data <- actimetrics::acti_count_data
fixed_window <- acti_sleep_setwindow(data, start_hour = 22, end_hour = 8)
```

---

acti_sleep_sib *Detect sustained inactivity bouts (SIBs)*

Description

For raw triaxial accelerometer data, the default `"vanHees2015"` method derives a longitudinal-axis angle at `epoch` resolution and identifies periods with no posture change above `angle_threshold`. For epoch-level data, provide a precomputed `sib` column; alternatively, `"low_activity"` marks epochs whose conventional activity metric is at or below `activity_threshold`.

Usage

```
acti_sleep_sib(
  data,
  method = c("vanHees2015", "low_activity"),
  epoch = "5 seconds",
  time_threshold_minutes = 5,
  angle_threshold = 5,
  activity_threshold = 0
)
```

Arguments

`data` Raw triaxial or regular epoch-level data.

`method` SIB method: `"vanHees2015"` or `"low_activity"`.

`epoch` Raw-data aggregation epoch.

`time_threshold_minutes`, `angle_threshold` van Hees posture-change parameters.

`activity_threshold` Low-activity cutoff for epoch data.

Value

A tibble with `time`, `sib`, and supporting epoch-level variables.

Examples

```
raw <- actiread::acti_read_gt3x(actiread::acti_example_gt3x(), verbose = FALSE)
sib <- acti_sleep_sib(raw)
```

---

acti_sleep_tudor_locke

Summarize sleep periods with Tudor–Locke

Description

Coerces actisomni-style epoch data to an `actigraph.sleepr` `tbl_agd` and delegates period detection and metric calculation to `actigraph.sleepr::apply_tudor_locke()`. This avoids maintaining a separate implementation of the Tudor–Locke rules.

Usage

```
acti_sleep_tudor_locke(
  data,
  time_col = "time",
  activity_col = NULL,
  sleep_col = "sleep",
  id_col = NULL,
  n_bedtime_start = 5,
  n_wake_time_end = 10,
  min_sleep_period = 160,
  max_sleep_period = 1440,
  min_nonzero_epochs = 0
)
```

Arguments

`data`	Epoch-level data with timestamp, activity-count, and sleep-label columns.
`time_col`	Name of the POSIXct timestamp column.
`activity_col`	Name of the non-negative activity-count column. When `NULL`, the first of `activity`, `counts`, `axis1`, or `count` is used.
`sleep_col`	Name of a logical sleep column or a character column using `S/W` or `sleep/wake` labels. Use `acti_sleep_tudor_locke_diary()` to create this column from a sleep diary.
`id_col`	Optional recording identifier column.
`n_bedtime_start`	Minimum consecutive sleep epochs required to begin a sleep period.

`n_wake_time_end`
: Wake epochs shorter than this are absorbed into sleep.

`min_sleep_period`
: Minimum retained sleep-period duration, in minutes.

`max_sleep_period`
: Maximum retained sleep-period duration, in minutes.

`min_nonzero_epochs`
: Minimum nonzero activity epochs in a retained period.

Value

A tibble of Tudor–Locke sleep periods and metrics, as returned by `actigraph.sleepr::apply_tudor_locke()`.

Examples

```
time <- as.POSIXct("2020-01-01", tz = "UTC") + 0:299 * 60
epochs <- data.frame(time = time, activity = 0, sleep = TRUE)
acti_sleep_tudor_locke(epochs)
```

---

acti_sleep_tudor_locke_diary

Apply a sleep diary to epoch data for Tudor–Locke

Description

Adds a logical sleep column to epoch data from diary onset and wakeup times. The resulting data can be passed directly to `acti_sleep_tudor_locke()`.

Usage

```
acti_sleep_tudor_locke_diary(
  data,
  sleep_diary,
  time_col = "time",
  diary_onset_col = "onset",
  diary_wakeup_col = "wakeup",
  sleep_col = "sleep"
)
```

Arguments

`data`
: Epoch-level data containing a POSIXct timestamp column.

`sleep_diary`
: A data frame with POSIXct sleep onset and wakeup columns. Each interval may span midnight; multiple days are supported.

`time_col`
: Name of the POSIXct timestamp column in `data`.

`diary_onset_col`
: Name of the diary onset column.

`diary_wakeup_col`
: Name of the diary wakeup column.

`sleep_col`
: Name of the logical sleep column to add or replace.

Value

`data` with a logical `sleep_col`, `TRUE` from onset (inclusive) to wakeup (exclusive).

`acti_sleeper` *Estimate sleep with the sleeper model*

Description

Runs `sleeper::estimate_sleep()` on complete, time-ordered raw triaxial acceleration. Input is converted to the timestamp-in-seconds format required by `sleeper` before the model is called.

Usage

```
acti_sleeper(data, epoch = 30L, model_dir)
```

Arguments

`data`
: A data frame containing timestamps and triaxial acceleration.

`epoch`
: Output epoch in seconds.

`model_dir`
: Directory containing the sleeper model files.

Value

An `acti_sleep_estimate` tibble with `time`, `sleep`, `sleep_probability`, `sleep_stage`, `nonwear`, and `method`. The sleeper backend does not return probabilities or sleep stages, so those columns are `NA`.

`py_acti_asleep` *Estimate sleep with asleep in an isolated Python process*

Description

Uses `asleep::py_asleep()` to run the same asleep model as `acti_asleep()` in a `callr` subprocess. This keeps Python package requirements separate from the current R session.

Usage

```
py_acti_asleep(
  data,
  min_wear_hours = 22L,
  time_shift = "0",
  report_light_and_temp = FALSE,
  pytorch_device = c("cpu", "cuda:0"),
  sample_rate = NULL,
  verbose = TRUE,
  force_download = FALSE,
  pyenv_function = function() asleep::py_require_asleep(),
  show = FALSE
)
```

Arguments

`data`	A data frame containing timestamps and triaxial acceleration, or a readable accelerometry file path. File paths are passed unchanged to `asleep::asleep()`.
`min_wear_hours`	Minimum daily wear time used by `asleep` summaries.
`time_shift`	Clock-time correction passed to `asleep::asleep()`.
`report_light_and_temp`	Whether to request light and temperature output.
`pytorch_device`	Device used for prediction.
`sample_rate`	Optional input sample rate.
`verbose`	Whether `asleep` prints progress messages.
`force_download`	Whether to re-download `asleep` model files.
`pyenv_function`	Function that configures Python in the subprocess.
`show`	Whether to stream subprocess output.

Value

An `acti_sleep_estimate` tibble; see `acti_asleep()`.

---

py_acti_sleeper *Estimate sleep with sleeper in an isolated Python process*

Description

Uses `sleeper::py_estimate_sleep()` to run the same sleeper model as `acti_sleeper()` in a `callr` subprocess.

Usage

```
py_acti_sleeper(
  data,
  epoch = 30L,
  model_dir,
  pyenv_function = function() sleeper::py_require_sleeper(),
  show = FALSE
)
```

Arguments

data	A data frame containing timestamps and triaxial acceleration.
epoch	Output epoch in seconds.
model_dir	Directory containing the sleeper model files.
pyenv_function	Function that configures Python in the subprocess.
show	Whether to stream subprocess output.

Value

An acti_sleep_estimate tibble; see acti_sleeper().

Index

acti_asleep, 2
acti_asleep(), *17, 18*
acti_calculate_asleep *(*acti_asleep*)*, 2
acti_sleep_diary, 4
acti_sleep_ensemble, 4
acti_sleep_fuse_labels, 5
acti_sleep_guider, 6
acti_sleep_guider_requirements, 7
acti_sleep_hdcza, 8
acti_sleep_hlrb, 9
acti_sleep_horangle, 9
acti_sleep_l5, 10
acti_sleep_label, 11
acti_sleep_labels, 12
acti_sleep_notworn, 13
acti_sleep_setwindow, 13
acti_sleep_sib, 14
acti_sleep_tudor_locke, 15
acti_sleep_tudor_locke(), *16*
acti_sleep_tudor_locke_diary, 16
acti_sleep_tudor_locke_diary(), *15*
acti_sleeper, 17
acti_sleeper(), *18, 19*
actigraph.sleepr::apply_tudor_locke(), *15, 16*
asleep::asleep(), *2, 3*, *18*
asleep::py_asleep(), *17*

lubridate::floor_date(), *8*, *10*

py_acti_asleep, 17
py_acti_sleeper, 18

sleeper::estimate_sleep(), *17*
sleeper::py_estimate_sleep(), *18*