

Package: actisomni (via r-universe)

October 17, 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 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](https://doi.org/10.1371/journal.pone.0142533)> and van Hees et al. (2018) <[doi:10.1038/s41598-018-31266-z](https://doi.org/10.1038/s41598-018-31266-z)>; the model estimators are described by Yuan et al. (2024) <[doi:10.1038/s41746-024-01065-0](https://doi.org/10.1038/s41746-024-01065-0)> and Sundararajan et al. (2021) <[doi:10.1038/s41598-020-79217-x](https://doi.org/10.1038/s41598-020-79217-x)>.

Roxygen list(markdown = TRUE)

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,

Config/roxygen2/version 8.0.0

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

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

Config/pak/sysreqs libicu-dev libpng-dev libssl-dev python3 libx11-dev

Repository <https://jhuwit.r-universe.dev>

Date/Publication 2026-10-15 18:45:11 UTC

RemoteUrl <https://github.com/jhuwit/actisomni>
RemoteRef HEAD
RemoteSha 638cd58d5584883a4d0f3d05e55740e462c70226

Contents

acti_asleep	2
acti_sleep_diary	3
acti_sleep_ensemble	4
acti_sleep_fuse_labels	5
acti_sleep_guider	6
acti_sleep_guider_requirements	7
acti_sleep_hdcza	7
acti_sleep_hlrb	8
acti_sleep_horangle	9
acti_sleep_l5	10
acti_sleep_label	11
acti_sleep_labels	11
acti_sleep_notworn	12
acti_sleep_setwindow	13
acti_sleep_sib	13
acti_sleeper	14
py_acti_asleep	15
py_acti_sleeper	16
Index	17

acti_asleep	<i>Estimate sleep with the asleep model</i>
-------------	---

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

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

<code>acti_sleep_diary</code>	<i>Identify a diary-defined sleep window</i>
-------------------------------	--

Description

Identify a diary-defined sleep window

Usage

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

Arguments

<code>data</code>	A data frame containing timestamps.
<code>onset, wakeup</code>	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	<i>Run a sleep-label ensemble from raw or epoch data</i>
---------------------	--

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	<i>Dispatch a sleep-period-time guider from data</i>
-------------------	--

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 <code>lubridate::floor_date()</code> .
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	<i>Identify the longest rest bout from sustained inactivity bouts</i>
-----------------	---

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	<i>Identify a sleep window from horizontal posture</i>
---------------------	--

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 <code>lubridate::floor_date()</code> .
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	<i>Identify a sleep window from the least-active five hours</i>
---------------	---

Description

Identify a sleep window from the least-active five hours

Usage

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

Arguments

<code>data</code>	A data frame of regular activity epochs.
<code>l5_hours</code>	Duration of the least-active period.
<code>window_hours</code>	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	<i>Convert a sleep guider to epoch-level sleep/wake labels</i>
------------------	--

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	<i>Create sleep/wake labels for multiple guiders</i>
-------------------	--

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	<i>Identify a low-activity window for no-night-wear protocols</i>
--------------------	---

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_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	<i>Estimate sleep with asleep in an isolated Python process</i>
----------------	---

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

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

Value

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

py_acti_sleeper	<i>Estimate sleep with sleeper in an isolated Python process</i>
-----------------	--

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(), [15](#)
acti_calculate_asleep (acti_asleep), [2](#)
acti_sleep_diary, [3](#)
acti_sleep_ensemble, [4](#)
acti_sleep_fuse_labels, [5](#)
acti_sleep_guider, [6](#)
acti_sleep_guider_requirements, [7](#)
acti_sleep_hdcza, [7](#)
acti_sleep_hlrb, [8](#)
acti_sleep_horangle, [9](#)
acti_sleep_l5, [10](#)
acti_sleep_label, [11](#)
acti_sleep_labels, [11](#)
acti_sleep_notworn, [12](#)
acti_sleep_setwindow, [13](#)
acti_sleep_sib, [13](#)
acti_sleeper, [14](#)
acti_sleeper(), [16](#)
asleep::asleep(), [2](#), [3](#), [15](#)
asleep::py_asleep(), [15](#)

lubridate::floor_date(), [8](#), [9](#)

py_acti_asleep, [15](#)
py_acti_sleeper, [16](#)

sleeper::estimate_sleep(), [14](#)
sleeper::py_estimate_sleep(), [16](#)