Package: actigraph.sleepr (via r-universe)

June 3, 2026

Type Package

Title Detect Periods of Sleep and Non-Wear in 'ActiGraph' Data

Version 0.3.1

Description Reads *.agd files exported from 'ActiGraph' devices; implements the Troiano (2008) <doi:10.1249/mss.0b013e31815a51b3> and Choi (2011) <doi:10.1249/MSS.0b013e3181ed61a3> algorithms for detecting periods on non-wear; implements the Sadeh (1994) <doi:10.1093/sleep/17.3.201> and Cole-Kripke (1992) <doi:10.1093/sleep/15.5.461> algorithms for detecting asleep/awake state and the Tudor-Locke (2014) <doi:10.1139/apnm-2013-0173> algorithm to detect sleep periods from asleep/awake states.

URL https://github.com/dipetkov/actigraph.sleepr

BugReports https://github.com/dipetkov/actigraph.sleepr/issues

License GPL (>= 2)

Encoding UTF-8

LazyData true

Roxygen list(markdown = TRUE)

RoxygenNote 8.0.0

Depends R (>= 3.2.4)

Imports DBI, RcppRoll, RSQLite, dplyr (>= 1.0.1), tidyr (>= 1.1.1), assertthat, ggplot2, lubridate, purrr, rlang, zoo, magrittr, tibble, tidyselect

Suggests covr, knitr, readr, rmarkdown, testthat, lintr

VignetteBuilder knitr

LinkingTo Rcpp

Config/pak/sysreqs libicu-dev

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

Date/Publication 2026-05-29 12:06:40 UTC

RemoteUrl https://github.com/dipetkov/actigraph.sleepr

RemoteRef HEAD

RemoteSha dcbfa3890f9bbfece87f97193f5caac7b88c9ed9

Contents

`actigraph.sleepr` *actigraph.sleepr*

Description

This package implements three standard algorithms for sleep detection from ActiGraph data: Sadeh, Cole-Kripke and Tudor-Locke.

Details

In addition to the help pages, see the README for examples.

Author(s)

Maintainer: Desislava Petkova <desislavka@gmail.com>

Authors:

- Desislava Petkova <desislavka@gmail.com>

Other contributors:

- John Muschelli [contributor]

See Also

Useful links:

- https://github.com/dipetkov/actigraph.sleepr
- Report bugs at https://github.com/dipetkov/actigraph.sleepr/issues

---

`apply_choi` *Apply the Choi algorithm*

Description

The Choi algorithm detects periods of non-wear in activity data from an ActiGraph device. Such intervals are likely to represent invalid data and therefore should be excluded from downstream analysis.

Usage

```
apply_choi(
  agdb,
  min_period_len = 90,
  min_window_len = 30,
  spike_tolerance = 2,
  use_magnitude = FALSE
)
```

Arguments

`agdb`	A `tibble` of activity data with an `epochlength` attribute. The epoch length must be 60 seconds.
`min_period_len`	Minimum number of consecutive "zero" epochs to start a non-wear period. The default is 90.
`min_window_len`	The minimum number of consecutive "zero" epochs immediately preceding and following a spike of artifactual movement. The default is 30.
`spike_tolerance`	Also known as artifactual movement interval. At most `spike_tolerance` "nonzero" epochs can occur in sequence during a non-wear period without interrupting it. The default is 2.

use_magnitude Logical. If true, the magnitude of the vector (axis1, axis2, axis3) is used to measure activity; otherwise the axis1 value is used. The default is FALSE.

Details

The Choi algorithm extends the Troiano algorithm by requiring that short spikes of artifactual movement during a non-wear period are preceded and followed by `min_window_len` consecutive "zero" epochs.

This implementation of the algorithm expects that the epochs are 60 second long.

Value

A summary `tibble` of the detected non-wear periods. If the activity data is grouped, then non-wear periods are detected separately for each group.

References

L Choi, Z Liu, CE Matthews and MS Buchowski. Validation of accelerometer wear and nonwear time classification algorithm. *Medicine & Science in Sports & Exercise*, 43(2):357–364, 2011.

ActiLife 6 User's Manual by the ActiGraph Software Department. 04/03/2012.

See Also

`apply_troiano()`, `collapse_epochs()`

Examples

```
library("dplyr")
data("gtxplus1day")

gtxplus1day %>%
  collapse_epochs(60) %>%
  apply_choi()
```

---

apply_cole_kripke *Apply the Cole-Kripke algorithm*

Description

The Cole-Kripke sleep scoring algorithm is primarily used for adult populations as the supporting research was performed on subjects ranging from 35 to 65 years of age.

Usage

```
apply_cole_kripke(agdb)
```

Arguments

agdb	A tibble of activity data with an epochlength attribute. The epoch length must be 60 seconds.

Details

The original paper proposes three versions of the Cole-Kripke algorithm, optimized for 1-minute, 30-second and 10-second epochs. Here only the 1-min version is implemented and therefore the apply_cole_kripke() function requires that the activity data is in 60s epochs. Use the collapse_epochs() function to modify higher-frequency data, if necessary.

The Cole-Kripke algorithm uses the y-axis (axis 1) counts. First epoch counts are divided by 100 and any scaled counts over 300 are clipped to 300. This transformation is specific to ActiGraph devices. The sleep index (SI) is defined as

```
.001 * (106 * epoch_prev(4) + 54 * epoch_prev(3) +
         58 * epoch_prev(2) + 76 * epoch_prev(1) +
        230 * epoch +
         74 * epoch_next(1) + 67 * epoch_next(2))
```

where at epoch t, epoch_prev(i) is the scaled activity count i epochs *before* t. Similarly, epoch_next(i) is the scaled activity count i epochs *after* t. That is, the algorithm uses a 7-epoch window which includes the four preceding and the two subsequent epochs. The time series of activity counts is padded with zeros as necessary, at the beginning and at the end.

Finally, the sleep state is awake (W) if the sleep index SI is less than 1; otherwise the sleep state is asleep (S).

Value

A tibble of activity data. A new column sleep indicates whether each 60s epoch is scored as asleep (S) or awake (W).

References

RJ Cole, DF Kripke, W Gruen, DJ Mullaney and JC Gillin. Automatic sleep/wake identification from wrist activity. *Sleep*, 15(5):461–469, 1992.

ActiLife 6 User's Manual by the ActiGraph Software Department. 04/03/2012.

See Also

collapse_epochs(), apply_sadeh(), apply_tudor_locke()

Examples

```
library("dplyr")
data("gtxplus1day")

gtxplus1day %>%
  collapse_epochs(60) %>%
  apply_cole_kripke()
```

`apply_sadeh` *Apply the Sadeh algorithm*

Description

The Sadeh sleep scoring algorithm is primarily used for younger adolescents as the supporting research was performed on children and young adults.

Usage

```
apply_sadeh(agdb)
```

Arguments

`agdb`	A `tibble` of activity data with an `epochlength` attribute. The epoch length must be 60 seconds.

Details

The Sadeh algorithm requires that the activity data is in 60s epochs and uses an 11-minute window that includes the five previous and five future epochs. This function implements the algorithm as described in the ActiGraph user manual.

The Sadeh algorithm uses the y-axis (axis 1) counts; epoch counts over 300 are set to 300. The sleep index (SI) is defined as

```
SI = 7.601 - (0.065 * AVG) - (1.08 * NATS) - (0.056 * SD) - (0.703 * LG)
```

where at epoch `t`

AVG the arithmetic mean (average) of the activity counts in an 11-epoch window centered at `t`

NATS the number of epochs in this 11-epoch window which have counts >= 50 and < 100

SD the standard deviation of the counts in a 6-epoch window that includes `t` and the five preceding epochs

LG the natural (base e) logarithm of the activity at epoch `t`. To avoid taking the log of 0, we add 1 to the count.

The time series of activity counts is padded with zeros as necessary, at the beginning and at the end, to compute the three functions AVG, SD, NATS within a rolling window.

Finally, the sleep state is asleep (S) if the sleep index SI is greater than -4; otherwise the sleep state is awake (W).

Value

A `tibble` of activity data. A new column `sleep` indicates whether each 60s epoch is scored as asleep (S) or awake (W).

References

A Sadeh, KM Sharkey and MA Carskadon. Activity based sleep-wake identification: An empirical test of methodological issues. *Sleep*, 17(3):201–207, 1994.

ActiLife 6 User's Manual by the ActiGraph Software Department. 04/03/2012.

See Also

`apply_cole_kripke()`, `apply_cole_kripke()`, `apply_tudor_locke()`

Examples

```
library("dplyr")
data("gtxplus1day")

gtxplus1day %>%
  collapse_epochs(60) %>%
  apply_sadeh()
```

---

apply_troiano *Apply the Troiano algorithm*

Description

The Troiano algorithm detects periods of non-wear in activity data from an ActiGraph device. Such intervals are likely to represent invalid data and therefore should be excluded from downstream analysis. The algorithm formalizes a technique used to analyze the 2003-2004 NHANES data; the original SAS source code can be found at https://riskfactor.cancer.gov/tools/nhanes_pam/.

Usage

```
apply_troiano(
  agdb,
  activity_threshold = 0,
  min_period_len = 60,
  max_nonzero_count = Inf,
  spike_tolerance = 2,
  spike_stoplevel = 100,
  use_magnitude = FALSE,
  endat_nnz_seq = TRUE
)
```

Arguments

agdb	A `tibble` of activity data with an `epochlength` attribute. The epoch length must be 60 seconds.

`activity_threshold`
: Highest activity level to be considered "zero"; an epoch with activity exceeding the threshold is considered a "spike". The default threshold is 0.

`min_period_len`
: Minimum number of consecutive "zero" epoch to start a non-wear period. The default is 60.

`max_nonzero_count`
: Epochs with activity greater than `max_nonzero_count` are labeled as "zero". The default is Inf.

`spike_tolerance`
: Also known as artifactual movement interval. At most `spike_tolerance` "nonzero" epochs can occur in sequence during a non-wear period without interrupting it. The default is 2.

`spike_stoplevel`
: An activity spike that exceeds `spike_stoplevel` counts ends a non-wear period, even if the spike tolerance has not been reached. The default is 100.

`use_magnitude`
: Logical. If true, the magnitude of the vector (axis1, axis2, axis3) is used to measure activity; otherwise the axis1 value is used. The default is FALSE.

`endat_nnz_seq`
: Logical. If true, a non-wear period ends with a run of nonzero epochs that is longer than `spike_tolerance`. This corresponds to the option *"Require consecutive epochs outside of the activity threshold"* in ActiLife's Wear Time Validation menu. The default is `TRUE`.

Details

The Troiano algorithm specifies that a non-wear period starts with `min_period_len` consecutive epochs/minutes of "zero" activity and ends with more than `spike_tolerance` epochs/minutes of "nonzero" activity.

This implementation of the algorithm expects 60s epochs.

Value

A summary `tibble` of the detected non-wear periods. If the activity data is grouped, then non-wear periods are detected separately for each group.

References

RP Troiano, D Berrigan, KW Dodd, LC Mâsse, T Tilert and M McDowell. Physical activity in the united states measured by accelerometer. *Medicine & Science in Sports & Exercise*, 40(1):181–188, 2008.

ActiLife 6 User's Manual by the ActiGraph Software Department. 04/03/2012.

See Also

`apply_choi()`, `collapse_epochs()`

Examples

```
library("dplyr")
data("gtxplus1day")

gtxplus1day %>%
  collapse_epochs(60) %>%
  apply_troiano()
```

`apply_tudor_locke` *Apply the Tudor-Locke algorithm*

Description

The Tudor-Locke algorithm detects periods of time in bed and, for each period, computes sleep quality metrics such as total minutes in bed, total sleep time, number and average length of awakenings, movement and fragmentation index.

Usage

```
apply_tudor_locke(
  agdb,
  n_bedtime_start = 5,
  n_wake_time_end = 10,
  min_sleep_period = 160,
  max_sleep_period = 1440,
  min_nonzero_epochs = 0
)
```

Arguments

`agdb`	A `tibble` of activity data with an `epochlength` attribute. The epoch length must be 60 seconds. Each epoch should be scored as asleep (S) or awake (W), using the Sadeh, the Cole-Kripke or a custom algorithm.
`n_bedtime_start`	Bedtime definition, in minutes. The default is 5.
`n_wake_time_end`	Wake time definition, in minutes. The default is 10.
`min_sleep_period`	Minimum sleep period length, in minutes. The default is 160.
`max_sleep_period`	Maximum sleep period length, in minutes. The default is 1440 (24 hours).
`min_nonzero_epochs`	Minimum number of epochs with non-zero activity. The default is 0.

Details

Once each one-minute epoch is labeled as asleep (S) or awake (W), we can use the Tudor-Locke algorithm to detect periods of *bedtime* and *sleep time*. By definition, *sleep time* < *bedtime* since one can be in bed and not sleeping.

Bedtime is (the first minute of) `n_bedtime_start` consecutive epochs/minutes labeled asleep (S). Similarly, wake time is (the first minute of) of `n_wake_time_end` consecutive epochs/minutes labeled awake (W), after a period of sleep. The block of time between bedtime and wake time is one sleep period, if the time elapsed is at least `min_sleep_period` minutes. There can be multiple sleep periods in 24 hours but a sleep period cannot be longer than `max_sleep_period` minutes.

For each sleep period, the algorithm calculates several measures of sleep quality such as time asleep and time awake, number and average length of awakenings, and movement and fragmentation indices.

This implementation of the Tudor-Locke algorithm detects all the sleep periods that ActiLife detects and, in some cases, it detects *additional* sleep periods. There are (at least) two such cases:

1. ActiLife filters out some sleep periods with *exactly* `min_nonzero_epochs` nonzero epochs. Alternatively, ActiLife computes the number of nonzero epochs in a sleep period differently and sometimes underestimates `nonzero_epochs` compared to `apply_tudor_locke()`.
2. ActiLife filters out sleep periods that end when the activity data ends, i.e., when the `out_bed_time` is also the final timestamp in the `agdb` table.

Value

A summary `tibble` of the detected sleep periods. If the activity data is grouped, then sleep periods are detected separately for each group.

in_bed_time The first minute of the bedtime.

out_bed_time The first minute of wake time.

onset The first minute that the algorithm scores "asleep".

latency The time elapsed between bedtime and sleep onset. By the definition of Tudor-Locke, latency is 0.

efficiency The number of sleep minutes divided by the bedtime minutes.

duration The duration of the sleep period, in minutes.

activity_counts The sum of activity counts for the entire sleep period.

total_sleep_time The number of minutes scored as "asleep" during the sleep period.

wake_after_onset The number of minutes scored as "awake", minus the sleep latency, during the sleep period.

nb_awakenings The number of awakening episodes.

ave_awakening The average length, in minutes, of all awakening episodes.

movement_index Proportion of awake time out of the total time in bed, in percentages.

fragmentation_index Proportion of one-minute sleep bouts out of the number of sleep bouts of any length, in percentages.

sleep_fragmentation_index The sum of the movement and fragmentation indices.

nonzero_epochs The number of epochs with activity > 0 (nonzero epochs).

References

C Tudor-Locke, TV Barreira, JM Schuna Jr, EF Mire and PT Katzmarzyk. Fully automated waist-worn accelerometer algorithm for detecting children's sleep-period time separate from 24-h physical activity or sedentary behaviors. *Applied Physiology*, Nutrition, and Metabolism, 39(1):53–57, 2014.

ActiLife 6 User's Manual by the ActiGraph Software Department. 04/03/2012.

See Also

`apply_sadeh()`, `apply_cole_kripke()`

Examples

```
library("dplyr")
library("lubridate")
data("gtxplus1day")

# Detect sleep periods using Sadeh as the sleep/awake algorithm
# and Tudor-Locke as the sleep period algorithm
agdb <- gtxplus1day %>%
  collapse_epochs(60) %>%
  filter(day(timestamp) == 28)
periods_sleep <- agdb %>%
  apply_sadeh() %>%
  apply_tudor_locke(min_sleep_period = 60)
periods_sleep

# Group and summarize by an extra varible (hour < 6 or not), which
# splits one long sleep period in two
agdb <- agdb %>%
  mutate(hour_lt6 = hour(timestamp) < 6) %>%
  group_by(hour_lt6)
periods_sleep <- agdb %>%
  apply_sadeh() %>%
  apply_tudor_locke(min_sleep_period = 60)
periods_sleep
```

---

collapse_epochs *Re-integrate epochs*

Description

Collapse post-filtered activity counts into larger epoch "buckets".

Usage

```
collapse_epochs(agdb, epoch_len_out, use_incomplete = TRUE)
```

Arguments

agdb	A tibble of activity data with an epochlength attribute.
epoch_len_out	Output (longer) epoch length in seconds, must be exact multiple of the input epoch length. Currently only epoch_len_out = 60 is supported.
use_incomplete	logical. Set to TRUE to follow ActiLife convention, which collapses all observed epochs even if they are incomplete.

Details

Activity counts cannot be reintegrated into shorter epochs, e.g., 60s -> 10s. Currently, collapse_epochs() integrates into 60s epochs only. This is not general but is sufficient for sleep analysis because the standard Sadeh and Cole-Kripke sleep algorithms were developed for 60s epoch data.

Suppose we want to collapse from 15 to 60 seconds. A complete 60s epoch consists of four 15s epochs: 00, 15, 45 and 60. However, the first and last epochs would be incomplete if the device started/stopped collecting data mid-minute. ActiLife 6 uses these epochs anyway. For example, if only 45 and 60 are available for the first minute, then ActiLife will aggregate across these two epochs only. This is a reasonable approach to sleep analysis with the Sadeh and the Cole-Kripke algorithms which pad the beginning and the end of the time series with zeros anyway.

Value

A tibble of activity data collapsed into one-minute epochs.

References

ActiLife 6 User's Manual by the ActiGraph Software Department. 04/03/2012.

Examples

```
library("dplyr")
data("gtxplus1day")

gtxplus1day %>%
  collapse_epochs(60)
```

---

combine_epochs_periods

Combine epochs with sleep/nonwear periods

Description

Suppose we have used apply_tudor_locke() to detect sleep periods or apply_troiano()/apply_choi() to detect non-wear periods. It might be useful to combine the epochs data with the periods data, so that each epoch is labeled according to which period it falls into, if any. Then we can easily slice the epochs data by sleep/non-sleep or wear/non-wear.

Usage

```
combine_epochs_periods(epochs, periods, start_var, end_var)
```

Arguments

epochs	A tibble of activity data with a timestamp column.
periods	A summary tibble of the (sleep or non-wear) periods.
start_var	The periods column which specifies when the periods start.
end_var	The periods column which specifies when the periods end.

Value

A tibble of activity data with one additional column, period_id, which indicates the period each epoch falls into.

Examples

```
library("dplyr")
data("gtxplus1day")

agdb <- gtxplus1day %>%
  collapse_epochs(60) %>%
  apply_sadeh()
periods <- agdb %>%
  apply_tudor_locke(min_sleep_period = 60)

agdb_with_periods <- combine_epochs_periods(
  agdb, periods,
  in_bed_time, out_bed_time
)

# How many sleep periods were detected and what is their duration,
# in minutes?
periods %>% select(in_bed_time, out_bed_time, duration)
# What is the assignment of epochs to periods?
agdb_with_periods %>% count(period_id)
```

---

complement_periods *Find the complement of time periods*

Description

Find the complement of a set of time periods in a set of epochs. For illustration, let's use integers instead of time periods and epochs. Suppose we have two intervals/periods, {[1,3], [8,10]}; their complement in the set {[0, ..., 12]} is {[0,0], [4,7], [11,12]}.

Usage

```
complement_periods(periods, epochs, start_var, end_var)
```

Arguments

periods	A tibble with at least two columns, start_var and end_var which are the first and the last epoch in a set of time periods, e.g. sleep periods or (non)wear periods.
epochs	A tibble with at least one column, timestamp, which contains POSIXct objects.
start_var	The variable (unquoted) which indicates when the time periods start.
end_var	The variable (unquoted) which indicates when the time periods end.

Value

A tibble of time periods with three columns: period_id (a sequential identifier), start_var (first epoch in period) and end_var (last epoch in period).

Examples

```
library("lubridate")
library("dplyr")
periods <- tibble(
  start = ymd_hm("2017-01-01 00:01"),
  end = ymd_hm("2017-01-01 00:05")
)
epochs <- tibble(timestamp = ymd_hm("2017-01-01 00:00") +
  minutes(0:12))
complement_periods(periods, epochs, start, end)
```

---

expand_periods *Expand time periods into a* tibble *of equally spaced time points*

Description

Expand time periods into a tibble of equally spaced time points

Usage

```
expand_periods(periods, start_var, end_var, units = "1 min")
```

Arguments

periods	A tibble with at least two columns, start_var and end_var which are the first and the last epoch in a set of time periods, e.g. sleep periods or (non)wear periods.
start_var	The variable (unquoted) which indicates when the time periods start.
end_var	The variable (unquoted) which indicates when the time periods end.
units	The time unit as a characters string. The default is "1 min".

Value

A tibble with two columns: period_id identifies the input period and timestamp is a POSIXct vector of time points within that period, equally spaced by units. Any grouping of periods is preserved.

Examples

```
library("dplyr")
data("gtxplus1day")

gtxplus1day %>%
  collapse_epochs(60) %>%
  apply_choi(min_period_len = 45) %>%
  expand_periods(period_start, period_end, units = "30 mins")
```

---

expand_timestamp *Expand a time period into a vector of equally spaced time points*

Description

Given the start time and the end time of a period, expand it into a vector of equally spaced time points.

Usage

```
expand_timestamp(start, end, units = "1 min")
```

Arguments

start	The start time, as a POSIXct object.
end	The end time, as a POSIXct object.
units	The time unit as a characters string. The default is "1 min".

Value

A POSIXct vector of equally spaced time points starting at start, ending no later than end, and separated by units.

Examples

```
start <- as.POSIXct("2017-01-01")
end <- as.POSIXct("2017-01-01 01:00:00")
expand_timestamp(start, end, "15 mins")
```

get_epoch_length *Guess the epoch length (in seconds) from the timestamp column*

Description

Guess the epoch length (in seconds) from the timestamp column

Usage

```
get_epoch_length(epochs)
```

Arguments

epochs	A tibble with at least one column, timestamp, which contains POSIXct objects.

Value

The epoch length in seconds, inferred from the spacing between consecutive timestamps. Throws an error if the time points are not equally spaced.

Examples

```
data("gtxplus1day")

gtxplus1day %>%
  get_epoch_length()

gtxplus1day %>%
  collapse_epochs(60) %>%
  get_epoch_length()
```

gtxplus1day *GT3X+ sample data*

Description

24 hours of actigraphy data collected with a GT3X+ device, which uses a tri-axial accelerometer to record activity in vertical (axis 1), horizontal (axis 2) and lateral (axis 3) directions.

Usage

```
gtxplus1day
```

Format

An object of class tbl_agd (inherits from tbl_df, tbl, data.frame) with 8999 rows and 4 columns.

has_missing_epochs *Checks whether there are gaps in the time series*

Description

The timestamps in the agd time series should run from `first(timestamp)` to `last(timestamp)` in increments of `epochlength` seconds. This function checks whether this holds or not. If the data is grouped (e.g., by subject), the check is performed for each group separately.

Usage

```
has_missing_epochs(agdb)
```

Arguments

`agdb`	A `tibble` of activity data with an `epochlength` attribute.

Value

`TRUE` or `FALSE`

impute_epochs *Impute missing count values*

Description

Trim leading and trailing NAs. Fill in the rest of the NAs using cubic spline interpolation.

Usage

```
impute_epochs(agdb, ...)
```

Arguments

`agdb`	A `tibble` of activity data with an `epochlength` attribute.
`...`	Comma separated list of unquoted variables.

Value

A `tibble` of activity data. Each variable in `...` is imputed.

See Also

`zoo::na.spline()`, `zoo::na.trim()`

Examples

```
library("dplyr")
data("gtxplus1day")

gtxplus1day$axis1[5:10] <- NA
gtxplus1day %>%
  impute_epochs(axis1)
```

`plot_activity` *Plot activity values*

Description

Plot a time series of activity values (by default, the counts on the vertical axis *axis1*).

Usage

```
plot_activity(agdb, var, color = "black", nrow = NULL, ncol = NULL)
```

Arguments

`agdb`	A `tibble` of activity data.
`var`	The activity variable (unquoted) to plot on the y-axis.
`color`	Activity line color.
`nrow, ncol`	Number of rows and columns. Relevant only if the activity data is grouped.

Value

A `ggplot` object. It shows the activity values in column `var` as a time series, faceted by group when the input `agdb` is grouped.

Examples

```
data("gtxplus1day")
sub_gt3x <- gtxplus1day %>%
  dplyr::filter(timestamp <= lubridate::as_datetime("2012-06-27 18:00:00"))

data <- sub_gt3x %>%
  collapse_epochs(60) %>%
  apply_cole_kripke()

plot_activity(data, axis1, color = "gray")
plot_activity(data, axis1, color = "sleep")
```

plot_activity_period *Plot activity and periods*

Description

Plot activity values as a time series and periods as polygons.

Usage

```
plot_activity_period(
  agdb,
  periods,
  act_var,
  start_var,
  end_var,
  color = "black",
  fill = "#525252",
  ncol = NULL,
  nrow = NULL
)
```

Arguments

agdb	A tibble of activity data.
periods	A tibble of periods with at least two columns start_var and end_var.
act_var	The activity variable (unquoted) to plot on the y-axis.
start_var	The variable (unquoted) which indicates when the time periods start.
end_var	The variable (unquoted) which indicates when the time periods end.
color	Activity line color.
fill	Polygon fill color.
nrow, ncol	Number of rows and columns. Relevant only if the activity data is grouped.

Value

A ggplot object. It shows the activity values as a time series and each period as a rectangle from start_var to end_var.

Examples

```
data("gtxplus1day")

sub_gt3x <- gtxplus1day %>%
  dplyr::filter(timestamp <= lubridate::as_datetime("2012-06-27 18:00:00"))
# Detect sleep periods using Sadeh as the sleep/awake algorithm
# and Tudor-Locke as the sleep period algorithm
periods_sleep <- sub_gt3x %>%
```

```
  collapse_epochs(60) %>%
  apply_cole_kripke() %>%
  apply_tudor_locke(min_sleep_period = 60)

plot_activity_period(
  sub_gt3x, periods_sleep, axis1,
  in_bed_time, out_bed_time
)
```

`read_agd` *Read activity counts from an *.agd file*

Description

Read ActiGraph sleep watch data from a database stored in an AGD file. Return a tibble.

Usage

```
read_agd(file, tz = "UTC")
```

Arguments

`file`	Full path to an agd file to read.
`tz`	Time zone to convert DateTime ticks to POSIX time.

Value

A `tibble` of activity data with at least two columns: timestamp and axis1 counts. Optional columns include axis2, axis2, steps, lux and inclinometer indicators (incline off, standing, sitting and lying). The device settings are stored as attributes, which include `epochlength`.

References

The AGD file format is described in the ActiLife 6 User's Manual by the ActiGraph Software Department, 04/03/2012 (Document SFT12DOC13, Revision A).

See Also

`read_agd_raw()`

Examples

```
file <- system.file("extdata", "GT3XPlus-RawData-Day01.agd",
  package = "actigraph.sleepr"
)
read_agd(file)

library("dplyr")
library("purrr")
```

```
# Read ActiGraph sleep watch data from the AGD files in a directory
# and bind the data into one data frame indexed by `.filename`.
path <- system.file("extdata", package = "actigraph.sleepr")

list.files(path, pattern = "*.agd", full.names = TRUE) %>%
  map_dfr(read_agd, .id = ".filename")
```

`read_agd_raw` *Read an *.agd file, with no post-processing*

Description

Read ActiGraph sleep watch data from an SQLite database stored in an AGD file and return a list with (at least) five tables: data, sleep, filters, settings, awakenings. The tables have the schema described in the ActiLife 6 User manual and the timestamps are converted from Unix time format to human-readable POSIXct representation.

Usage

```
read_agd_raw(file, tz = "UTC")
```

Arguments

`file`	Full path to an agd file to read.
`tz`	Time zone to convert DateTime ticks to POSIX time.

Details

Some ActiGraph devices contain a capacitive sensor to detect monitor removal when worn against the skin. If that data is available, the return list includes a capsense table as well.

Value

A list of five tables: settings, data, filters, sleep, awakenings and, if available, capsense.

References

ActiLife 6 User's Manual by the ActiGraph Software Department, 04/03/2012 (Document SFT12DOC13, Revision A).

`covertagd`: R package for converting agd files from ActiGraph into data.frames.

See Also

`read_agd()`

Examples

```
file <- system.file("extdata", "GT3XPlus-RawData-Day01.agd",
  package = "actigraph.sleepr"
)
str(read_agd_raw(file))
```

---

tbl_agd *A* tibble *of activity data exported by an ActiGraph device*

Description

This tibble has several attributes, most importantly, epochlength.

Usage

```
tbl_agd(data, settings)
```

Arguments

data	A tibble of raw activity counts.
settings	A tibble of device settings.

Value

A tibble of activity data. It has a time index, timestamp, one or more activity value columns, and attributes holding settings such as epochlength.

Examples

```
data("gtxplus1day")
gtxplus1day
attr(gtxplus1day, "epochlength")
```

Index