Package 'actiRhythm'

July 29, 2026

Type Package

Title Circadian Rest-Activity Rhythm Analysis

Version 0.1.0

Date 2026-06-21

Description Quantifies the circadian rest-activity rhythm, the roughly 24-hour cycle of activity and rest, from activity counts and raw accelerometer recordings. Analyses run on an activity-count vector and its timestamps, and a built-in reader loads 'ActiGraph' '.agd' files (an 'SQLite' database) directly. Computes nonparametric metrics (interdaily stability (IS) and intradaily variability (IV) following Witting et al. (1990) <doi:10.1016/0006-3223(90)90523-5>; relative amplitude (RA), the least-active 5-hour window (L5), and the most-active 10-hour window (M10) following Van Someren et al. (1999) <doi:10.3109/07420529908998724>), cosinor models with a rhythmicity test, confidence ellipses, population-mean cosinor, and two-group parameter comparison (Cornelissen (2014) <doi:10.1186/1742-4682-11-16>; Marler et al. (2006) <doi:10.1002/sim.2466>), period estimation by Lomb-Scargle (Lomb (1976) <doi:10.1007/BF00648343>; Scargle (1982) <doi:10.1086/160554>) and chi-square (Sokolove and Bushell (1978) <doi:10.1016/0022-5193(78)90022-X>) periodograms with bootstrap confidence intervals and a sliding-window spectrogram, fractal and nonlinear measures (Peng et al. (1994) <doi:10.1103/PhysRevE.49.1685>; Kantelhardt et al. (2002) <doi:10.1016/S0378-4371(02)01383-3>), the Sleep Regularity Index (Phillips et al. (2017) <doi:10.1038/s41598-017-03171-4>), social jet lag, rest-activity state transitions, and locomotor inactivity during sleep (Winnebeck et al. (2018) <doi:10.1016/j.cub.2017.11.063>). It also reads raw accelerometer files ('ActiGraph' '.gt3x', 'Axivity' '.cwa', 'GENEActiv' '.bin'), auto-calibrates them (van Hees et al. (2014) <doi:10.1152/japplphysiol.00421.2014>), and derives the ENMO, MAD, and z-angle metrics with diary-free sleep-period detection (van Hees et al. (2018) <doi:10.1038/s41598-018-31266-z>), cross-checked against 'GGIR'. A batch mode runs the analysis over a folder of files and writes a multi-sheet 'Excel' workbook.

License MIT + file LICENSE

URL https://github.com/rdazadda/actiRhythm,

https://rdazadda.github.io/actiRhythm/

BugReports https://github.com/rdazadda/actiRhythm/issues

Language en-US

Depends R (>= 4.1)

Imports DBI, grDevices, ggplot2, Rcpp (>= 1.0.0), Rdpack, RSQLite, scales, splines, stats, utils

LinkingTo Rcpp

RdMacros Rdpack

Suggests ActCR, agcounts, cosinor, cosinor2, GGIR, GGIRread, knitr, nonlinearTseries, nparACT, openxlsx, pracma, read.gt3x, rmarkdown, testthat (>= 3.0.0)

SystemRequirements C++17

VignetteBuilder knitr

Encoding UTF-8

Config/testthat/edition 3

Config/roxygen2/version 8.0.0

NeedsCompilation yes

Author Raymond Dacosta Azadda [aut, cre], CEAL Team [aut], Stacy Rasmus [aut]

Maintainer Raymond Dacosta Azadda <rdazadda@alaska.edu>

Repository CRAN

Date/Publication 2026-07-29 16:40:34 UTC

Contents

`actiRhythm_colors` *actiRhythm Color Generator*

Description

Generates colors from the actiRhythm palettes for visualizations. These colors match the CSS design system.

Usage

```
actiRhythm_colors(n = NULL, type = "categorical")
```

Arguments

`n`	Integer. Number of colors to return. If NULL, returns all.
`type`	Character. One of "categorical", "sequential", "diverging", or "intensity"

Value

A character vector of hex color codes

Examples

```
actiRhythm_colors(4)

library(ggplot2)
ggplot(mtcars, aes(wt, mpg, color = factor(cyl))) +
  geom_point() +
  scale_color_manual(values = actiRhythm_colors(3))
```

`activity.balance.index` *Activity Balance Index*

Description

The Activity Balance Index (Danilevicz et al. 2024), a 0 to 1 transform of a detrended fluctuation analysis scaling exponent that peaks at the healthy $\alpha = 1$ (1/f) balance: $ABI(\alpha) = \exp(-|\alpha - 1|/e^{-2}) = \exp(-e^2\,|\alpha - 1|)$.

Usage

```
activity.balance.index(x)
```

Arguments

`x`	Either a numeric scaling exponent, or a fractal object with `alpha` (and optionally `alpha1`, `alpha2`) such as the result of the package's detrended fluctuation analysis.

Value

If `x` is numeric, the scalar ABI. If `x` is a fractal object, a list with `ABI_overall`, `ABI_short`, `ABI_long`.

References

Danilevicz IM, van Hees VT, van der Heide F, Jacob L, Landre B, Benadjaoud MA, Sabia S (2024). "Measures of fragmentation of rest activity patterns: mathematical properties and interpretability based on accelerometer real life data." *BMC Medical Research Methodology*, **24**, 132. doi:10.1186/s1287402402255w.

Examples

```
activity.balance.index(1.0)   # perfect 1/f balance -> 1
activity.balance.index(0.7)
```

`activity.extrema` *Generalized Least- and Most-Active Periods (MX / LX)*

Description

The least-active `LX` and most-active `MX` periods for arbitrary window lengths, generalizing L5 and M10 (Van Someren et al. 1999). For each window the function returns the mean activity over the window and its onset and midpoint clock times. A fixed L5/M10 pair does not give these phase markers.

Usage

```
activity.extrema(counts, timestamps, windows = c(5, 10))
```

Arguments

`counts`	Numeric activity vector (minute epochs recommended).
`timestamps`	POSIXct timestamps, one per value.
`windows`	Window lengths in hours (default `c(5, 10)` = L5, M10).

Value

An object of class `actiRhythm_mxlx`: a data frame with, for each window, the least- and most-active mean level, onset hour, and midpoint hour. Never errors.

References

Van Someren EJW, Swaab DF, Colenda CC, Cohen W, McCall WV, Rosenquist PB (1999). "Bright light therapy: improved sensitivity to its effects on rest-activity rhythms in Alzheimer patients by application of nonparametric methods." *Chronobiology International*, **16**(4), 505–518. doi:10.3109/07420529908998724.

Examples

```
ts <- seq(as.POSIXct("2024-01-01", tz = "UTC"), by = 60, length.out = 4 * 1440)
h  <- as.numeric(format(ts, "%H"))
activity.extrema(ifelse(h >= 23 | h < 7, 5, 300), ts, windows = c(5, 8, 10))
```

activity.onset.offset *Activity Onset and Offset (Relative-Difference Phase Markers)*

Description

Finds the daily activity onset and offset by a relative-difference contrast on the averaged 24-hour profile: the onset is where mean activity rises most sharply (the relative difference of the window after versus before is largest) and the offset is where it falls most sharply. These are non-cosinor, non-changepoint phase markers, a normalized-contrast edge detector on the daily profile rather than a published actigraphy algorithm.

Usage

```
activity.onset.offset(counts, timestamps, window_hours = 6)
```

Arguments

counts	Numeric activity vector.
timestamps	POSIXct timestamps, one per value.
window_hours	Half-window, in hours, compared before versus after each minute (default 6).

Value

An object of class actiRhythm_aont: onset_h and offset_h (clock hours) and the relative-difference profile. Never errors.

Examples

```
ts <- seq(as.POSIXct("2024-01-01", tz = "UTC"), by = 60, length.out = 4 * 1440)
h  <- as.numeric(format(ts, "%H"))
activity.onset.offset(ifelse(h >= 23 | h < 7, 5, 300), ts)
```

agd.counts *Extract Counts from .agd Data*

Description

Extracts activity counts, timestamps, steps, and inclinometer data (if available) from AGD file data. Inclinometer data can be used to enhance sleep detection for AGD files that lack raw acceleration.

Usage

```
agd.counts(agd_data, convert.timestamps = TRUE, include_inclinometer = TRUE)
```

Arguments

`agd_data`	List returned from read.agd()
`convert.timestamps`	Logical. Convert ActiGraph timestamps to POSIXct (default: TRUE)
`include_inclinometer`	Logical. Include inclinometer data if available? (default: TRUE)

Details

Inclinometer data is particularly useful for sleep detection in AGD files:

- `inclineLying` = 1 indicates the device detected lying posture
- Combined with low activity (Cole-Kripke sleep), this sharpens specificity
- Not all AGD files contain inclinometer data (depends on ActiLife processing settings)

Value

Data frame with counts per minute, timestamps, and optional inclinometer data:

- `timestamp` - POSIXct timestamp (if convert.timestamps = TRUE)
- `axis1` - Vertical axis counts (Y-axis)
- `axis2` - Horizontal axis counts (X-axis)
- `axis3` - Anterior-posterior axis counts (Z-axis)
- `steps` - Step counts
- `lux` - Light intensity (if available)
- `inclineOff` - Off-body indicator (if available)
- `inclineStanding` - Standing indicator (if available)
- `inclineSitting` - Sitting indicator (if available)
- `inclineLying` - Lying indicator (if available)

`auto.calibrate` *Auto-Calibrate Raw Acceleration to the Unit Gravity Sphere*

Description

Estimates per-axis gain and offset corrections by the van Hees et al. (2014) method: during non-movement windows the acceleration vector should lie on the 1 g sphere, so gain/offset are fit by iteratively projecting non-movement window means onto the closest point of the unit sphere. Apply as `(raw - offset) * scale`.

Usage

```
auto.calibrate(
  xyz,
  fs,
  sphere_crit = 0.3,
  sd_crit = 0.013,
  max_iter = 1000,
  tol = 1e-10
)
```

Arguments

xyz	A data frame or matrix of raw acceleration in g (columns x, y, z).
fs	Sample rate in Hz.
sphere_crit	Minimum coverage (g) each axis must span on both sides of zero for a stable fit (default 0.3).
sd_crit	Per-axis SD (g) over consecutive (non-overlapping) 10-second windows below which a window counts as non-movement (default 0.013).
max_iter	Maximum refinement iterations (default 1000).
tol	Convergence tolerance on the calibration error (default 1e-10).

Value

A list with `scale` and `offset` (length-3), the calibration error before and after (`cal_error_start`, `cal_error_end`, mean absolute deviation from 1 g), `npoints`, a `calibrated` flag, and a `reason` string describing the outcome. When there is too little non-movement data the identity correction is returned with `calibrated = FALSE`.

References

van Hees VT, Fang Z, Langford J, Assah F, Mohammad A, da Silva ICM, Trenell MI, White T, Wareham NJ, Brage S (2014). "Autocalibration of accelerometer data for free-living physical activity assessment using local gravity and temperature: an evaluation on four continents." *Journal of Applied Physiology*, **117**(7), 738–744. doi:10.1152/japplphysiol.00421.2014.

Examples

```
# Recover a known per-axis gain and offset from non-movement windows
set.seed(1)
u <- matrix(rnorm(40 * 3), 40, 3); u <- u / sqrt(rowSums(u^2))   # sphere directions
raw <- do.call(rbind, lapply(seq_len(40), function(i)
  matrix(rep(u[i, ] / c(1.03, 0.97, 1.01) + c(0.04, -0.03, 0.02), each = 300),
         300, 3) + rnorm(900, 0, 0.004)))
auto.calibrate(data.frame(x = raw[, 1], y = raw[, 2], z = raw[, 3]), fs = 30)$scale
```

`axivity.counts` *Activity Counts from a Raw Axivity .cwa File*

Description

Reads a raw Axivity (`.cwa`) accelerometer file and converts it to ActiGraph-equivalent activity counts via the agcounts implementation of the ActiGraph count algorithm (Neishabouri 2022). Requires the **GGIRread** and **agcounts** packages.

Usage

```
axivity.counts(path, epoch = 60, lfe = FALSE, tz = "UTC")
```

Arguments

`path`	Path to a `.cwa` file.
`epoch`	Epoch length in seconds (default 60).
`lfe`	Use the low-frequency extension filter (default `FALSE`).
`tz`	Time zone for the timestamps (default `"UTC"`).

Value

Data frame with `time`, `axis1`, `axis2`, `axis3` and `vm`, one row per epoch (the same shape as `gt3x.counts`).

Cross-brand counts

These are an *approximation* of ActiGraph counts, not native ActiGraph output. Axivity-to-count conversion has been directly validated (Brond et al. 2017). The result is appropriate for the relative and normalized circadian metrics in this package (IS, IV, RA, L5, M10, SRI); it should *not* be used to apply ActiGraph intensity cut-points or to compare absolute counts across device brands.

References

Neishabouri A, Nguyen J, Samuelsson J, Guthrie T, Biggs M, Wyatt J, Cross D, Karas M, Migueles JH, Khan S, Guo CC (2022). "Quantification of acceleration as activity counts in ActiGraph wearable." *Scientific Reports*, **12**, 11958. doi:10.1038/s41598022160003x.

Brond JC, Andersen LB, Arvidsson D (2017). "Generating ActiGraph counts from raw acceleration recorded by an alternative monitor." *Medicine & Science in Sports & Exercise*, **49**(11), 2351–2360. doi:10.1249/MSS.0000000000001344.

See Also

`geneactiv.counts`, `gt3x.counts`, `read.raw`

backend_info *Backend Information*

Description

Backend Information

Usage

```
backend_info()
```

Value

Invisible list with backend info

chi.sq.periodogram *Chi-square (Sokolove-Bushell) Periodogram*

Description

Estimates the dominant period of an activity time series with the Sokolove-Bushell (1978) chi-square periodogram, the periodogram most widely used in chronobiology and actigraphy. It pairs with the Lomb-Scargle estimator in `circadian.period`: Lomb-Scargle is the least-squares spectral method for unevenly sampled data; the chi-square periodogram is the analysis-of-variance method for regularly sampled data and reports a significance threshold.

Usage

```
chi.sq.periodogram(
  counts,
  timestamps,
  from = 18,
  to = 30,
  alpha = 0.05,
  epoch_length = NULL
)
```

Arguments

`counts`	Numeric vector of activity counts on a regular epoch grid. `NA` values (e.g. non-wear) are handled by removal from the phase and grand means.
`timestamps`	A `POSIXct` vector (or numeric seconds) the same length as `counts`; used to infer the epoch length.
`from, to`	Period search window in hours (default `18` to `30`).
`alpha`	Significance level for the chi-square threshold (default `0.05`).
`epoch_length`	Epoch length in seconds. If `NULL` (default) it is inferred from the median spacing of `timestamps`.

Details

For each trial period P (an integer number of epochs) the series is folded into P phase bins over $K = \lfloor N/P \rfloor$ complete cycles and the statistic

$$Q_P = \frac{K\,N \sum_{h=1}^{P} (\bar{A}_h - \bar{A})^2}{\sum_{i=1}^{N} (A_i - \bar{A})^2}$$

is computed, where $\bar{A}_h$ is the mean at phase h and $\bar{A}$ the grand mean over the $N = KP$ retained points. Under the null hypothesis of no rhythm at P, Q_P follows a chi-square distribution with $P - 1$ degrees of freedom, giving the significance line $\chi^2_{P-1,\,1-\alpha}$. The estimated period is the P maximising Q_P within the search window.

Value

A named list with `period` (hours, the Q_P peak), `Qp_peak`, `p_value` (family-wise Sidak p-value of the peak across the scanned periods), `significant` (logical, `p_value < alpha`), `scanned` (trial periods in hours), `Qp` (the periodogram aligned to `scanned`), `critical` (the per-period chi-square threshold), `epoch_length` and `alpha`. On insufficient/degenerate data the same shape is returned with `period`/`Qp_peak` `NA` and empty vectors; the function never throws.

References

Sokolove PG, Bushell WN (1978). "The chi square periodogram: its utility for analysis of circadian rhythms." *Journal of Theoretical Biology*, **72**(1), 131–160. doi:10.1016/00225193(78)90022X.

Refinetti R, Cornelissen G, Halberg F (2007). "Procedures for numerical analysis of circadian rhythms." *Biological Rhythm Research*, **38**(4), 275–325. doi:10.1080/09291010600903692.

Sidak Z (1967). "Rectangular confidence regions for the means of multivariate normal distributions." *Journal of the American Statistical Association*, **62**(318), 626–633. doi:10.1080/01621459.1967.10482935.

See Also

`circadian.period` for the Lomb-Scargle estimator.

`circadian-analysis` *Circadian Rhythm Analysis*

Description

Non-parametric methods for characterizing 24-hour activity patterns.

References

Witting W, Kwa IH, Eikelenboom P, Mirmiran M, Swaab DF (1990). "Alterations in the circadian rest-activity rhythm in aging and Alzheimer's disease." *Biological Psychiatry*, **27**(6), 563–572. doi:10.1016/00063223(90)905235.

Van Someren EJW, Swaab DF, Colenda CC, Cohen W, McCall WV, Rosenquist PB (1999). "Bright light therapy: improved sensitivity to its effects on rest-activity rhythms in Alzheimer patients by application of nonparametric methods." *Chronobiology International*, **16**(4), 505–518. doi:10.3109/07420529908998724.

Phillips AJK, Clerx WM, O'Brien CS, Sano A, Barger LK, Picard RW, Lockley SW, Klerman EB, Czeisler CA (2017). "Irregular sleep/wake patterns are associated with poorer academic performance and delayed circadian and sleep/wake timing." *Scientific Reports*, **7**(1), 3216. doi:10.1038/s41598017031714.

Wittmann M, Dinich J, Merrow M, Roenneberg T (2006). "Social jetlag: misalignment of biological and social time." *Chronobiology International*, **23**(1-2), 497–509. doi:10.1080/07420520500545979.

`circadian-fractal` *Fractal and Complexity Metrics for Activity Time Series*

Description

Two nonlinear complexity and scaling metrics for accelerometer activity time series: Detrended Fluctuation Analysis (DFA) and Multiscale Sample Entropy (MSE). Both run in base R (only **stats** is used) so they add no new runtime dependency. They measure the temporal structure (long-range correlations and information-theoretic complexity) of minute-level activity counts, alongside the amplitude- and timing-based circadian metrics in `circadian.rhythm`.

References

Peng CK, Buldyrev SV, Havlin S, Simons M, Stanley HE, Goldberger AL (1994). "Mosaic organization of DNA nucleotides." *Physical Review E*, **49**(2), 1685–1689. doi:10.1103/PhysRevE.49.1685.

Hu K, Ivanov PC, Chen Z, Carpena P, Stanley HE (2001). "Effect of trends on detrended fluctuation analysis." *Physical Review E*, **64**(1), 011114. doi:10.1103/PhysRevE.64.011114.

Hu K, Van Someren EJW, Shea SA, Scheer FAJL (2009). "Reduction of scale invariance of activity fluctuations with aging and Alzheimer's disease: Involvement of the circadian pacemaker." *Proceedings of the National Academy of Sciences*, **106**(8), 2490–2494. doi:10.1073/pnas.0806087106.

Richman JS, Moorman JR (2000). "Physiological time-series analysis using approximate entropy and sample entropy." *American Journal of Physiology - Heart and Circulatory Physiology*, **278**(6), H2039–H2049. doi:10.1152/ajpheart.2000.278.6.H2039.

Costa M, Goldberger AL, Peng C (2002). "Multiscale entropy analysis of complex physiologic time series." *Physical Review Letters*, **89**(6), 068102. doi:10.1103/PhysRevLett.89.068102.

Costa M, Goldberger AL, Peng C (2005). "Multiscale entropy analysis of biological signals." *Physical Review E*, **71**(2), 021906. doi:10.1103/PhysRevE.71.021906.

`circadian-period` *Endogenous Circadian Period Estimation (Lomb-Scargle)*

Description

Estimates the dominant (endogenous) circadian PERIOD (tau) of an activity time series using the Lomb-Scargle periodogram. Unlike the classical Fast Fourier Transform (FFT), the Lomb-Scargle method does not require evenly sampled data, so it correctly handles the irregular and gappy sampling that results from non-wear periods, dropped epochs, or mixed epoch lengths in accelerometer recordings.

`circadian-plots` *Circadian Rhythm Visualizations*

Description

ggplot2 visualizations for the circadian / chronobiology metrics produced by actiRhythm. These functions are thin plotting wrappers around the existing analytic engines and add no new fitting logic:

- `plot_periodogram` draws the Lomb-Scargle periodogram and highlights the endogenous period estimated by `circadian.period`.
- `plot_extended_cosinor` overlays the Marler extended (anti-logistic) cosinor fit from `cosinor.antilogistic` and the ordinary cosinor from `cosinor.analysis` on the averaged 24-hour activity profile.
- `plot_dfa` draws the detrended-fluctuation log-log scaling plot from `fractal.dfa`.

Every function returns a `ggplot` object, never errors on degenerate or insufficient input (it returns an annotated empty plot instead), and falls back to `ggplot2::theme_minimal()` when the package theme helpers are unavailable.

References

Lomb NR (1976). "Least-squares frequency analysis of unequally spaced data." *Astrophysics and Space Science*, **39**(2), 447–462. doi:10.1007/BF00648343.

Scargle JD (1982). "Studies in astronomical time series analysis. II. Statistical aspects of spectral analysis of unevenly spaced data." *The Astrophysical Journal*, **263**, 835–853. doi:10.1086/160554.

Marler MR, Gehrman P, Martin JL, Ancoli-Israel S (2006). "The sigmoidally transformed cosine curve: a mathematical model for circadian rhythms with symmetric non-sinusoidal shapes." *Statistics in Medicine*, **25**(22), 3893–3904. doi:10.1002/sim.2466.

Peng CK, Buldyrev SV, Havlin S, Simons M, Stanley HE, Goldberger AL (1994). "Mosaic organization of DNA nucleotides." *Physical Review E*, **49**(2), 1685–1689. doi:10.1103/PhysRevE.49.1685.

`circadian.batch` *Batch Circadian Analysis of Multiple AGD Files*

Description

Reads a set of ActiGraph `.agd` files, runs the full circadian analysis on each, and returns one combined summary row per file, the package equivalent of uploading a batch of files. Files that fail to read or analyse are reported in an `error` column rather than aborting the run. Optionally writes a single Excel workbook whose Summary sheet has one row per file.

Usage

```
circadian.batch(
  files,
  file = NULL,
  metric = c("axis1", "vm"),
  include_period_ci = FALSE,
  n_boot = 200,
  epoch_length = NULL,
  verbose = TRUE
)
```

Arguments

`files`	Character vector of `.agd` paths, or a single directory (all its `.agd` files are used).
`file`	Optional output `.xlsx` path for a combined-summary workbook.
`metric`	Activity metric: `"axis1"` (default) or `"vm"` (vector magnitude).
`include_period_ci`	Bootstrap the period CI per file (default `FALSE` for batch speed).
`n_boot`	Bootstrap replicates when `include_period_ci` is `TRUE`.
`epoch_length`	Epoch length in seconds; `NULL` (default) infers it per file from the timestamps.
`verbose`	Print per-file progress (default `TRUE`).

Value

A data frame with one row per file (a `file` column, an `error` column, and every summary metric). Returned invisibly when a workbook `file` is written.

See Also

`circadian.workbook`

Examples

```
# Pass a folder to analyse a whole batch; a single file is used here.
batch <- circadian.batch(example_agd(), verbose = FALSE)
batch[, c("file", "IS", "IV", "RA", "rhythm_p_value")]
```

`circadian.daily` *Per-Day Nonparametric Metrics*

Description

The intraday nonparametric metrics computed for each day separately, showing within-recording drift that the pooled values hide. Interdaily stability is a between-day measure and so is not a per-day quantity; the per-day table reports L5, M10, their onset times, relative amplitude, and intradaily variability (Goncalves et al. 2014).

Usage

```
circadian.daily(counts, timestamps, min_hours = 12)
```

Arguments

`counts`	Numeric activity vector (minute epochs recommended).
`timestamps`	POSIXct timestamps, one per value.
`min_hours`	Minimum hours of data a day needs to be reported (default 12).

Value

An object of class `actiRhythm_daily`: a `daily` data frame, one row per day. Never errors.

References

Goncalves BSB, Cavalcanti PRA, Tavares GR, Campos TF, Araujo JF (2014). "Nonparametric methods in actigraphy: an update." *Sleep Science*, **7**(3), 158–164. doi:10.1016/j.slsci.2014.09.013.

Examples

```
ts <- seq(as.POSIXct("2024-01-01", tz = "UTC"), by = 60, length.out = 4 * 1440)
h  <- as.numeric(format(ts, "%H"))
circadian.daily(ifelse(h >= 23 | h < 7, 5, 300), ts)
```

`circadian.emd` *Empirical Mode Decomposition of the Activity Rhythm*

Description

Decomposes the activity series into intrinsic mode functions by empirical mode decomposition, a data-adaptive, nonlinear alternative to the linear SSA decomposition (Huang et al. 1998). The intrinsic mode function whose period is nearest 24 hours is taken as the circadian component. Optional ensemble EMD adds noise to reduce mode mixing (Wu and Huang 2009).

Usage

```
circadian.emd(
  counts,
  timestamps,
  max_imf = 10L,
  ensemble = 1L,
  noise_sd = 0.2,
  period_range = c(20, 28),
  epoch_length = 60,
  seed = NULL
)
```

Arguments

`counts`	Numeric activity vector (a coarse epoch is recommended for speed).
`timestamps`	POSIXct timestamps, one per value.
`max_imf`	Maximum number of modes to extract (default 10).
`ensemble`	Ensemble size for EEMD (default 1 = plain EMD).
`noise_sd`	Added-noise SD as a fraction of the series SD (default 0.2).
`period_range`	Period window (hours) for the circadian mode (default 20 to 28).
`epoch_length`	Epoch length in seconds (default 60).
`seed`	Optional seed for the EEMD noise.

Value

An object of class `actiRhythm_emd`: the IMF matrix, the residual trend, per-IMF period and variance share, the circadian IMF index, and the reconstruction error. The first and last epochs are unreliable (the spline envelopes pin the IMFs near zero at the edges), so read the instantaneous series away from the boundary. Never errors.

References

Huang NE, Shen Z, Long SR, Wu MC, Shih HH, Zheng Q, Yen NC, Tung CC, Liu HH (1998). "The empirical mode decomposition and the Hilbert spectrum for nonlinear and non-stationary time series analysis." *Proceedings of the Royal Society A*, **454**(1971), 903–995. doi:10.1098/rspa.1998.0193.

Wu Z, Huang NE (2009). "Ensemble empirical mode decomposition: a noise-assisted data analysis method." *Advances in Adaptive Data Analysis*, **1**(1), 1–41. doi:10.1142/S1793536909000047.

See Also

`hilbert.huang`, `circadian.ssa`

Examples

```
ts <- seq(as.POSIXct("2024-01-01", tz = "UTC"), by = 600, length.out = 6 * 144)
th <- as.numeric(difftime(ts, ts[1], units = "hours"))
circadian.emd(100 + 60 * cos(2 * pi * th / 24), ts, epoch_length = 600)
```

---

circadian.flm *Functional Linear Model of the 24-Hour Activity Profile*

Description

Fits the averaged 24-hour activity profile with a periodic basis expansion (Fourier by default, B-spline alternative) by weighted least squares, giving a smooth functional form of the daily activity pattern. The single-component cosinor is the one-harmonic special case; adding harmonics fits the non-sinusoidal shape of a real rest-activity profile. This follows the functional form of Wang et al. (2011), as implemented for actigraphy by pyActigraphy.

Usage

```
circadian.flm(
  counts,
  timestamps,
  basis = c("fourier", "bspline"),
  n_harmonics = 4,
  nbasis = 9,
  spline_order = 4,
  period = 24,
  wear_time = NULL,
  min_valid_hours = 10,
  weights = c("n", "none"),
  n_eval = 1440
)
```

Arguments

counts	Numeric activity vector.
timestamps	POSIXct timestamps, one per value.
basis	Basis type: "fourier" (default) or "bspline".
n_harmonics	Number of Fourier harmonics (default 4, giving a 9-term expansion as in Wang et al. (2011)).
nbasis	Number of B-spline basis functions (default 9).
spline_order	B-spline order (default 4, cubic).
period	Profile period in hours (default 24).
wear_time	Optional logical wear-time mask.
min_valid_hours	Minimum valid hours per day (default 10).
weights	Profile weighting: "n" (square root of the per-hour observation count, default, matching cosinor.analysis) or "none" (plain least squares, matching pyActigraphy).
n_eval	Length of the dense within-day evaluation grid (default 1440).

Value

An object of class actiRhythm_flm: the fitted coefficients, the smooth daily curve (smooth_curve), the fitted profile, per-harmonic amplitudes and acrophases (harmonics, Fourier only), the peak and trough, and the fit statistics (r_squared, aic, f_statistic, p_value). The function never errors; on insufficient data it returns the same structure with r_squared NA.

References

Wang J, Xian H, Licis A, Deych E, Ding J, McLeland J, Toedebusch C, Li T, Duntley S, Shannon W (2011). "Measuring the impact of apnea and obesity on circadian activity patterns using functional linear modeling of actigraphy data." *Journal of Circadian Rhythms*, **9**, 11. doi:10.1186/17403391-911.

Ramsay JO, Silverman BW (2005). *Functional Data Analysis*, 2nd edition. Springer. doi:10.1007/b98888.

Hammad G, Reyt M, Beliy N, Baillet M, Deantoni M, Lesoinne A, Muto V, Schmidt C (2021). "pyActigraphy: open-source python package for actigraphy data visualization and analysis." *PLOS Computational Biology*, **17**(10), e1009514. doi:10.1371/journal.pcbi.1009514.

See Also

cosinor.analysis, circadian.ssa

Examples

```
ts <- seq(as.POSIXct("2024-01-01", tz = "UTC"), by = 60, length.out = 4 * 1440)
h  <- as.numeric(format(ts, "%H")) + as.numeric(format(ts, "%M")) / 60
counts <- 100 + 60 * cos(2 * pi * (h - 14) / 24) + 25 * cos(2 * pi * (h - 6) / 12)
circadian.flm(counts, ts)
```

`circadian.is.multiscale` *Multi-resolution Interdaily Stability (IS)*

Description

Computes interdaily stability at several within-day bin resolutions. IS is classically reported at 60-min bins, but finer bins (30/15 min) capture higher-frequency day-to-day regularity. For each bin width, `IS = (N * sum_h (xbar_h - xbar)^2) / (p * sum_i (x_i - xbar)^2)` where `p` is bins-per-day, `xbar_h` the mean for bin-of-day `h`, and `N` the total number of bins.

Usage

```
circadian.is.multiscale(
  counts,
  timestamps,
  bin_minutes = (1:60)[1440L%%(1:60) == 0L]
)
```

Arguments

`counts`	Numeric vector of epoch-level activity.
`timestamps`	POSIXct timestamps (one per epoch).
`bin_minutes`	Integer bin widths in minutes that divide 1440 (default the divisors of 1440 from 1 to 60 min, per Goncalves et al. 2014).

Value

An object of class `actiRhythm_ism`: a per-bin `IS` table and the averaged `ISm`.

References

Witting W, Kwa IH, Eikelenboom P, Mirmiran M, Swaab DF (1990). "Alterations in the circadian rest-activity rhythm in aging and Alzheimer's disease." *Biological Psychiatry*, **27**(6), 563–572. doi:10.1016/00063223(90)905235.

Goncalves BSB, Cavalcanti PRA, Tavares GR, Campos TF, Araujo JF (2014). "Nonparametric methods in actigraphy: an update." *Sleep Science*, **7**(3), 158–164. doi:10.1016/j.slsci.2014.09.013.

circadian.onset.ci *Confidence Intervals for L5/M10 Onset Timing*

Description

Percentile bootstrap confidence interval for the mean daily onset time of a circadian phase marker (e.g. L5 or M10), using circular resampling of the per-day onsets.

Usage

```
circadian.onset.ci(onset_hours, level = 0.95, n_boot = 2000)
```

Arguments

onset_hours	Numeric vector of daily onset times in decimal hours.
level	Confidence level (default 0.95).
n_boot	Bootstrap replicates (default 2000).

Value

List with mean_onset, ci_lower, ci_upper (decimal hours) and n_days.

circadian.period *Estimate Endogenous Circadian Period via the Lomb-Scargle Periodogram*

Description

Computes the Lomb-Scargle periodogram of an activity counts series sampled at the supplied timestamps and returns the period (tau, in hours) of the strongest spectral peak within the search window [from, to]. The Lomb-Scargle method (Lomb 1976; Scargle 1982) is the least-squares-equivalent spectral estimator for unevenly sampled time series and is therefore appropriate for actigraphy data containing gaps, which the FFT cannot accommodate.

Usage

```
circadian.period(counts, timestamps, from = 18, to = 30, ofac = 4)
```

Arguments

counts	Numeric vector of activity counts (minute-level recommended). NA values (e.g. non-wear epochs) are dropped together with their timestamps before estimation.
timestamps	A POSIXct vector (or anything coercible by as.numeric) of epoch timestamps, the same length as counts. Internally converted to hours elapsed since the first timestamp.
from	Numeric. Lower bound of the period search window, in hours (default 18).
to	Numeric. Upper bound of the period search window, in hours (default 30).
ofac	Integer oversampling factor controlling the period-grid resolution. Higher values give a finer period grid and a more precise peak location at the cost of computation (default 4).

Details

Processing steps:

1. Timestamps are converted to hours since the first sample (t_hours = (as.numeric(timestamps) - min) / 3600).
2. Pairs with a missing count or a missing/non-finite time are dropped.
3. Two guards are applied so the estimate is never based on insufficient data: the recording must span at least **2 days** (otherwise a 18-30 h period cannot be resolved) and at least **10 non-**NA observations must remain.
4. The standard-normalized Lomb-Scargle periodogram is evaluated over the period window. Its strongest peak gives the period (hours), and the Baluev (2008) analytic false-alarm probability gives the p-value.

The Lomb-Scargle periodogram is chosen specifically because actigraphy series are rarely gap-free: the FFT assumes uniform sampling, whereas Lomb-Scargle fits sinusoids by least squares at each trial frequency and is unbiased for irregular sampling.

Value

A named list with elements:

tau Numeric. Period (hours) of the strongest Lomb-Scargle peak in [from, to], i.e. the estimated endogenous circadian period. NA_real_ when the data are insufficient.

peak_power Numeric. Normalized power of that peak (the Lomb-Scargle peak statistic). NA_real_ when insufficient.

p_value Numeric. P-value of the peak under the null hypothesis of Gaussian noise (Baluev 2008 analytic false-alarm probability). NA_real_ when insufficient.

oversampling The ofac oversampling factor used.

n_used Integer. Number of non-NA observations actually passed to the periodogram (NA_integer_ when not run).

span_days Numeric. Total recording span in days (max minus min timestamp), used for the >= 2-day guard.

scanned Numeric vector of trial periods (hours) of the full Lomb-Scargle spectrum (`numeric(0)` when not run).

power Numeric vector of standard-normalized Lomb-Scargle power, aligned to `scanned` (`numeric(0)` when not run).

On any edge case (too few points, too short a span, degenerate input, or an internal numerical failure) the function returns this same structure with `tau`, `peak_power` and `p_value` set to `NA`; it never throws.

References

Lomb NR (1976). "Least-squares frequency analysis of unequally spaced data." *Astrophysics and Space Science*, **39**(2), 447–462. doi:10.1007/BF00648343.

Scargle JD (1982). "Studies in astronomical time series analysis. II. Statistical aspects of spectral analysis of unevenly spaced data." *The Astrophysical Journal*, **263**, 835–853. doi:10.1086/160554.

Ruf T (1999). "The Lomb-Scargle periodogram in biological rhythm research: analysis of incomplete and unequally spaced time-series." *Biological Rhythm Research*, **30**(2), 178–201. doi:10.1076/brhm.30.2.178.1422.

Baluev RV (2008). "Assessing the statistical significance of periodogram peaks." *Monthly Notices of the Royal Astronomical Society*, **385**(3), 1279–1285. doi:10.1111/j.13652966.2008.12689.x.

Refinetti R, Cornelissen G, Halberg F (2007). "Procedures for numerical analysis of circadian rhythms." *Biological Rhythm Research*, **38**(4), 275–325. doi:10.1080/09291010600903692.

See Also

`cosinor.analysis` for parametric (fixed-period) rhythm estimation, `circadian.rhythm` for non-parametric L5/M10/IS/IV metrics.

Examples

```
# Seven days of minute-level data with a 24 h rhythm
t_hours <- seq(0, 7 * 24 - 1/60, by = 1/60)
ts <- as.POSIXct("2024-01-01 00:00:00") + t_hours * 3600
counts <- 100 + 80 * cos(2 * pi * (t_hours - 8) / 24) + rnorm(length(t_hours), 0, 5)
circadian.period(counts, ts)$tau   # about 24.0
```

`circadian.quotient` *Circadian Quotient and Relative Amplitude from a Cosinor Fit*

Description

Computes the circadian quotient (amplitude divided by MESOR) and a cosinor relative amplitude (amplitude divided by the overall mean) from a cosinor result. The circadian quotient is a dimensionless measure of rhythm strength against the rhythm-adjusted mean.

Usage

```
circadian.quotient(cosinor_result)
```

Arguments

`cosinor_result`	A list returned by `cosinor.analysis()` (class `"actiRhythm_cosinor"`) or any list providing `amplitude` and `mesor`. An optional `overall_mean` (raw arithmetic mean of the counts) is used for the relative-amplitude denominator when present; otherwise `mesor` is used.

Value

A list with class `"actiRhythm_circadian_quotient"` containing:

circadian_quotient `amplitude / mesor`.

relative_amplitude `amplitude / overall_mean` (falls back to `amplitude / mesor` when `overall_mean` is absent).

Returns `NA` entries when the inputs are missing or the denominator is non-positive.

References

Nelson W, Tong YL, Lee JK, Halberg F (1979). "Methods for cosinor-rhythmometry." *Chronobiologia*, **6**(4), 305–323.

Examples

```
ts <- seq(as.POSIXct("2024-01-01", tz = "UTC"), by = 60, length.out = 1440 * 3)
counts <- 100 + 80 * cos(2 * pi * (as.numeric(format(ts, "%H")) - 14) / 24)
cos <- cosinor.analysis(counts, ts)
circadian.quotient(cos)
```

`circadian.raw` *Circadian Analysis Directly from Raw Acceleration*

Description

From a raw accelerometer file (or raw data frame), computes a per-epoch raw metric (ENMO or MAD) with auto-calibration, then runs the count-style circadian analysis (`circadian.rhythm`) on it. The metrics (IS, IV, RA, L5/M10, cosinor, DFA, periodograms, SRI, ...) run on the raw metric the same way they run on counts, so raw data needs no new per-method code.

Usage

```
circadian.raw(
  x,
  metric = c("ENMO", "MAD"),
  device = "auto",
  epoch = 60,
  calibrate = TRUE,
  tz = "UTC",
  ...
)
```

Arguments

x	A path to a raw file (.gt3x, .cwa, .bin) or a raw data frame (see raw.metrics / example_raw).
metric	Raw metric to analyse, "ENMO" (default) or "MAD".
device	Device brand or "auto" (default; used for file input).
epoch	Epoch length in seconds (default 60).
calibrate	Apply van Hees auto-calibration first (default TRUE).
tz	Time zone (default "UTC").
...	Passed to circadian.rhythm.

Value

The circadian.rhythm result computed on the raw metric.

See Also

raw.metrics, circadian.rhythm, example_raw

Examples

```
# The full count-style battery on synthetic raw ENMO

cr <- circadian.raw(example_raw(days = 2), metric = "ENMO")
c(IS = cr$IS, IV = cr$IV, RA = cr$RA)
```

`circadian.rhythm` *Circadian Rhythm Analysis*

Description

Computes non-parametric circadian metrics as in GGIR, ActCR, and nparACT. Uses a minute-level sliding window for L5/M10, plus the Sleep Regularity Index and several phase metrics.

Usage

```
circadian.rhythm(
  counts,
  timestamps,
  sleep_state = NULL,
  wear_time = NULL,
  min_valid_hours = 10,
  epoch_length = 60,
  calculate_sri = TRUE,
  use_cpp = TRUE
)
```

Arguments

`counts`	Numeric vector of activity counts (minute-level recommended for accuracy)
`timestamps`	POSIXct vector of epoch timestamps
`sleep_state`	Optional character vector of sleep states ("S" or "W") for SRI calculation
`wear_time`	Optional logical vector indicating wear time (TRUE = worn)
`min_valid_hours`	Numeric. Valid-day criterion (GGIR includedaycrit): minimum wear hours for a day to count. Applied only when `wear_time` is given; default `10`. Set `0`/`NULL` to disable.
`epoch_length`	Numeric. Epoch length in seconds (default: 60)
`calculate_sri`	Logical. Calculate Sleep Regularity Index? (default: TRUE if sleep_state provided)
`use_cpp`	Logical. Use C++ backend for faster computation? (default: TRUE)

Details

Non-Parametric Metrics (L5/M10/RA: van Someren et al., 1999; IS/IV: Witting et al., 1990):

L5/M10 use a **minute-level sliding window**, not hourly aggregation, so timing is resolved to the minute rather than the hour.

- **L5** = Average activity during the least active 5 consecutive hours. Uses sliding window across minute-level data for precise timing.
- **M10** = Average activity during the most active 10 consecutive hours.

- **RA** = Relative amplitude = (M10 - L5) / (M10 + L5). Ranges 0 to 1; higher values mean stronger day-night amplitude.
- **IS** = Interdaily stability. Measures coupling of the rhythm to stable zeitgebers. Range 0 (Gaussian noise) to 1 (perfect stability).
- **IV** = Intradaily variability. Measures rhythm fragmentation. About 0 for a perfect sine wave, about 2 for Gaussian noise, and can exceed 2 with ultradian rhythms.

Sleep Regularity Index (Phillips et al., 2017):

SRI = probability of being in same sleep/wake state at any two time points 24 hours apart. Range -100 to +100, with 100 indicating perfect regularity.

Value

List with class 'actiRhythm_circadian' containing:

L5, L5_start, L5_start_hour Least active 5-hour average and timing (sliding window)

M10, M10_start, M10_start_hour Most active 10-hour average and timing (sliding window)

L1, M1 Least/most active 1-hour for additional granularity

RA Relative amplitude: (M10-L5)/(M10+L5), range 0-1

IS Interdaily stability: day-to-day consistency (0-1, higher=more stable)

IV Intradaily variability: within-day fragmentation (0 for a sine wave, near 2 for noise)

phi First-order autocorrelation at 1-hour lag

SRI Sleep Regularity Index (-100 to 100, higher=more regular)

onset_timing_variability Mean circular SD of daily L5/M10 onset times (day-to-day phase variability). NOT the published Fischer/Roenneberg CPD.

hourly_profile Mean activity by hour of day

daily_metrics Per-day L5, M10, RA values

See Also

`sleep.regularity.index` for standalone SRI calculation, `social.jet.lag` for social jet lag calculation

Examples

```
counts <- agd.counts(read.agd(example_agd()))
result <- circadian.rhythm(counts$axis1, counts$timestamp)
print(result)
```

`circadian.spectrogram` *Circadian Spectrogram (Period over Time)*

Description

Slides a window across the recording and computes the chi-square (Sokolove-Bushell) periodogram in each window, producing a period-by-time map that shows how the dominant period and its strength drift across the recording (non-stationarity, fragmentation, re-entrainment). A single global periodogram or cosinor fit cannot show this.

Usage

```
circadian.spectrogram(
  counts,
  timestamps,
  window_hours = 72,
  step_hours = 6,
  from = 18,
  to = 30,
  epoch_length = NULL
)
```

Arguments

`counts`	Numeric activity vector.
`timestamps`	POSIXct timestamps, one per value.
`window_hours`	Sliding-window length in hours (default 72).
`step_hours`	Step between successive windows in hours (default 6).
`from`, `to`	Period search window in hours (default 18, 30).
`epoch_length`	Epoch length in seconds (default 60).

Value

An object of class `actiRhythm_spectrogram`: a long `data` frame (window centre time, period, power) and a `ggplot` heat map in `$plot`.

See Also

`chi.sq.periodogram`, `circadian.period`

Examples

```
t_hours <- seq(0, 8 * 24, by = 1 / 60)
ts <- as.POSIXct("2024-01-01", tz = "UTC") + t_hours * 3600
counts <- 100 + 80 * cos(2 * pi * t_hours / 24) + rnorm(length(t_hours), 0, 20)
circadian.spectrogram(counts, ts, step_hours = 24)$plot
```

`circadian.ssa` *Singular Spectrum Analysis of an Activity Series*

Description

Decomposes an activity-count series into additive components (trend, a circadian component, ultradian components, and noise) with Basic Singular Spectrum Analysis: embed the series into a Hankel trajectory matrix, take its singular value decomposition, group the resulting elementary series, and reconstruct each group by diagonal averaging. pyActigraphy implements this model-free decomposition for actigraphy.

Usage

```
circadian.ssa(
  counts,
  timestamps,
  window_hours = 24,
  n_components = 10,
  groups = NULL,
  w_components = NULL,
  period_range = c(20, 28),
  detrend = FALSE
)
```

Arguments

`counts`	Numeric activity vector.
`timestamps`	POSIXct timestamps, one per value.
`window_hours`	Embedding window length in hours (default 24). The window in epochs is `L = round(window_hours * 3600 / epoch_seconds)`.
`n_components`	Number of leading elementary components to reconstruct and keep (default 10).
`groups`	Optional named list of 1-based component indices, e.g. `list(trend = 1, circadian = 2:3)`. When `NULL` the grouping is chosen automatically (component 1 is the trend; the circadian pair is the largest component whose period falls in `period_range`).
`w_components`	Number of leading components used for the w-correlation matrix (default `min(n_components, 10)`).
`period_range`	Two-element period window in hours used to identify the circadian component (default `c(20, 28)`).
`detrend`	If `TRUE`, the trend component is removed from `reconstructed`.

Details

Singular Spectrum Analysis on a long minute-level series builds a large trajectory matrix; for multi-day recordings, resampling to a coarser epoch (for example 10 to 30 minutes) before calling keeps the decomposition fast.

Value

An object of class `actiRhythm_ssa`: the singular values and partial variances, the reconstructed component series (`trend`, `circadian`, `ultradian`), the w-correlation matrix, the circadian `fundamental_period`, and the share of variance the circadian component carries. The function never errors; on insufficient data it returns the same structure with `insufficient = TRUE`.

References

Golyandina N, Zhigljavsky A (2013). *Singular Spectrum Analysis for Time Series*, SpringerBriefs in Statistics. Springer. doi:10.1007/9783642349133.

Vautard R, Yiou P, Ghil M (1992). "Singular-spectrum analysis: a toolkit for short, noisy chaotic signals." *Physica D: Nonlinear Phenomena*, **58**(1-4), 95–126. doi:10.1016/01672789(92)90103T.

Hammad G, Reyt M, Beliy N, Baillet M, Deantoni M, Lesoinne A, Muto V, Schmidt C (2021). "pyActigraphy: open-source python package for actigraphy data visualization and analysis." *PLOS Computational Biology*, **17**(10), e1009514. doi:10.1371/journal.pcbi.1009514.

See Also

`circadian.period`, `circadian.flm`

Examples

```
ts <- seq(as.POSIXct("2024-01-01", tz = "UTC"), by = 1800, length.out = 7 * 48)
h  <- as.numeric(format(ts, "%H")) + as.numeric(format(ts, "%M")) / 60
counts <- 50 + 0.5 * seq_along(ts) + 60 * cos(2 * pi * (h - 14) / 24)
ssa <- circadian.ssa(counts, ts)
ssa
```

`circadian.wavelet` *Continuous Wavelet Transform of the Activity Rhythm*

Description

Runs a Morlet continuous wavelet transform on the activity series and returns the time-frequency power surface across circadian and ultradian periods and a dominant-period-over-time track (Torrence and Compo 1998; Leise 2013). Unlike the sliding chi-square spectrogram, it localizes period drift at every time point, so a lengthening or fragmenting rhythm shows up directly.

Usage

```
circadian.wavelet(counts, timestamps, dj = 1/12, omega0 = 6, epoch_length = 60)
```

Arguments

counts	Numeric activity vector (a coarse epoch, e.g. 10-minute bins, is recommended for speed; see the example).
timestamps	POSIXct timestamps, one per value.
dj	Scale resolution in voices per octave step (default 1/12).
omega0	Morlet central frequency (default 6).
epoch_length	Epoch length in seconds (default 60).

Value

An object of class `actiRhythm_wavelet`: the period grid (hours), the time-averaged global power spectrum, the per-time dominant period, the overall peak period, the power matrix, and the cone of influence (`coi_period_h`, the largest reliable period at each time, with the `in_coi` mask of edge-affected cells). The global spectrum and dominant period are computed outside the cone so edge effects do not bias them. A `significant` logical matrix flags cells whose power exceeds the 95\ confidence level against an AR(1) red-noise background (with the per-scale threshold `sig_power` and the lag-1 autocorrelation `phi`), which separates a real rhythm from background. The power is scale-rectified ($|W|^2/s$, Liu et al. 2007), which reduces the Torrence bias toward long periods. Never errors.

References

Torrence C, Compo GP (1998). "A practical guide to wavelet analysis." *Bulletin of the American Meteorological Society*, **79**(1), 61–78. doi:10.1175/15200477(1998)079<0061:APGTWA>2.0.CO;2.

Leise TL (2013). "Wavelet analysis of circadian and ultradian behavioral rhythms." *Journal of Circadian Rhythms*, **11**, 5. doi:10.1186/17403391115.

Liu Y, Liang XS, Weisberg RH (2007). "Rectification of the bias in the wavelet power spectrum." *Journal of Atmospheric and Oceanic Technology*, **24**(12), 2093–2102. doi:10.1175/2007JTECHO511.1.

Examples

```
ts <- seq(as.POSIXct("2024-01-01", tz = "UTC"), by = 600, length.out = 6 * 144)
th <- as.numeric(difftime(ts, ts[1], units = "hours"))
circadian.wavelet(100 + 80 * cos(2 * pi * th / 24), ts, epoch_length = 600)
```

---

circadian.workbook *Export a Full Circadian Analysis to an Excel Workbook*

Description

Runs the complete actiRhythm circadian analysis on one activity time series and writes a multi-sheet `.xlsx` workbook: a one-row **Summary** of every metric (nonparametric IS/IV/RA/L5/M10, cosinor, the rhythmicity F-test, the Lomb-Scargle period with its bootstrap confidence interval, the chi-square periodogram, DFA, multifractal DFA, multiscale entropy, rest-activity state transitions, and, when sleep is supplied, the Sleep Regularity Index, social jet lag and LIDS), plus detail sheets for the hourly profile, both periodograms, the fluctuation and multifractal spectra, the transition curves, the LIDS fits, and a data dictionary.

Usage

```
circadian.workbook(
  activity,
  timestamps,
  file = NULL,
  sleep_state = NULL,
  sleep_periods = NULL,
  wear_time = NULL,
  epoch_length = 60,
  include_period_ci = TRUE,
  n_boot = 200
)
```

Arguments

`activity`	Numeric activity vector.
`timestamps`	POSIXct timestamps, one per value.
`file`	Output `.xlsx` path; if `NULL` the workbook object is returned without writing.
`sleep_state`	Optional per-epoch sleep/wake vector (turns on the SRI).
`sleep_periods`	Optional sleep-period data frame with `in_bed_time`/`out_bed_time` (turns on social jet lag and LIDS).
`wear_time`	Optional logical wear-time mask.
`epoch_length`	Epoch length in seconds (default 60).
`include_period_ci`	Whether to bootstrap the period CI (default TRUE).
`n_boot`	Bootstrap replicates for the period CI (default 200).

Value

The `openxlsx` workbook object, invisibly.

Examples

```
ts <- seq(as.POSIXct("2024-01-01", tz = "UTC"), by = 60, length.out = 2 * 1440)
h  <- as.numeric(format(ts, "%H"))
act <- pmax(0, 100 + 80 * cos(2 * pi * (h - 14) / 24) + rnorm(length(ts), 0, 20))
circadian.workbook(act, ts, file = tempfile(fileext = ".xlsx"), include_period_ci = FALSE)
```

circadian_cpp *Complete Circadian Analysis*

Description

Complete Circadian Analysis

Usage

```
circadian_cpp(minute_data, hours_per_day = 24L, start_minute = 0L)
```

Arguments

minute_data	Numeric vector of minute-level activity
hours_per_day	Hours per day (default: 24)
start_minute	Start minute of day (0-1439)

Value

List with L5, M10, RA, IS, IV, phi

composite.phase.deviation *Composite Phase Deviation (Fischer & Roenneberg, 2016)*

Description

Combines, for each day, the deviation of the phase marker from the individual's own mean phase (precision) and from a reference phase (accuracy) as `CPD = mean(sqrt(precision^2 + accuracy^2))`. When no external reference phase is supplied, accuracy is measured from the individual's own mean phase. CPD is distinct from the circular SD reported by `onset_timing_variability`.

Usage

```
composite.phase.deviation(onset_hours, reference_phase = NULL)
```

Arguments

`onset_hours`	Numeric vector of daily phase-marker onset times in decimal hours (e.g. daily L5 onsets).
`reference_phase`	Optional reference phase in decimal hours (e.g. a scheduled/expected time, or a group mean). With the default `NULL` the accuracy term is measured from the individual's own mean phase, so it collapses onto the precision term and `CPD = sqrt(2) * mean(|precision|)`, a within-individual dispersion measure. The published misalignment CPD of Fischer and Roenneberg requires an external `reference_phase`.

Value

List with `CPD`, `precision` (mean absolute deviation from own mean phase, hours), `accuracy` (mean absolute deviation from the reference, hours), `reference_phase`, and `n_days`.

References

Fischer D, Vetter C, Roenneberg T (2016). "A novel method to visualise and quantify circadian misalignment." *Scientific Reports*, **6**, 38601. doi:10.1038/srep38601.

`consensus.rhythmicity` *Consensus Rhythmicity Across Methods*

Description

Combines the rhythmicity verdicts of three tests into one call: the cosinor zero-amplitude F-test (`rhythmicity.test`), the Lomb-Scargle Baluev false-alarm probability (`circadian.period`), and the chi-square (Sokolove-Bushell) periodogram (`chi.sq.periodogram`). The three share one series, so their p-values are pooled by the Cauchy combination, which is valid under arbitrary dependence, and a majority vote is also reported.

Usage

```
consensus.rhythmicity(
  counts,
  timestamps,
  period = 24,
  alpha = 0.05,
  wear_time = NULL
)
```

Arguments

counts	Numeric activity vector.
timestamps	POSIXct timestamps, one per value.
period	Rhythm period in hours for the cosinor tests (default 24).
alpha	Significance level (default 0.05).
wear_time	Optional logical wear-time mask for the cosinor fit.

Value

An object of class actiRhythm_consensus: the combined p-value and consensus call, the vote count, a per-method tests data frame, and the agreement fraction.

References

Liu Y, Xie J (2020). "Cauchy combination test: a powerful test with analytic p-value calculation under arbitrary dependency structures." *Journal of the American Statistical Association*, **115**(529), 393–402. doi:10.1080/01621459.2018.1554485.

Examples

```
ts <- seq(as.POSIXct("2024-01-01", tz = "UTC"), by = 60, length.out = 3 * 1440)
h <- as.numeric(format(ts, "%H"))
counts <- pmax(0, 100 + 80 * cos(2 * pi * (h - 14) / 24))
consensus.rhythmicity(counts, ts)
```

---

cosinor.analysis *Cosinor Analysis for Circadian Rhythm*

Description

Fits a single-component cosinor model to activity data and returns the parametric circadian parameters.

Usage

```
cosinor.analysis(
  counts,
  timestamps,
  period = 24,
  wear_time = NULL,
  min_valid_hours = 10,
  transform = c("none", "log1p")
)
```

Arguments

`counts`	Numeric vector of activity counts
`timestamps`	POSIXct vector of timestamps
`period`	Period in hours (default: 24 for circadian rhythm)
`wear_time`	Optional logical vector indicating wear time
`min_valid_hours`	Numeric. Valid-day criterion (GGIR includedaycrit): minimum wear hours for a day to count; default 10, 0/NULL to disable.
`transform`	Input transform: `"none"` (default) or `"log1p"`, GGIR's `log(ENMO_mg + 1)` pre-transform for raw acceleration metrics (g-unit input is scaled to mg first).

Details

The single-component cosinor model fits:

$$Y(t) = M + A \cdot cos(2\pi t/T + \phi)$$

Where:

- M = mesor (rhythm-adjusted mean)
- A = amplitude (half peak-to-trough difference)
- T = period (typically 24 hours)
- ϕ = acrophase (phase angle, converted to time)

The model is fit using linear least squares on the linearized form:

$$Y(t) = M + \beta_1 \cos(2\pi t/T) + \beta_2 \sin(2\pi t/T)$$

Value

List with class 'actiRhythm_cosinor' containing:

mesor Rhythm-adjusted mean (MESOR)

amplitude Half the peak-to-trough difference

acrophase Time of peak activity (hours, 0-24)

acrophase_time Acrophase as HH:MM string

r_squared Goodness of fit (coefficient of determination)

f_statistic F-statistic for model significance

p_value P-value for model significance

References

Nelson W, Tong YL, Lee JK, Halberg F (1979). "Methods for cosinor-rhythmometry." *Chronobiologia*, **6**(4), 305–323.

Cornelissen G (2014). "Cosinor-based rhythmometry." *Theoretical Biology and Medical Modelling*, **11**, 16. doi:10.1186/174246821116.

Examples

```
counts <- agd.counts(read.agd(example_agd()))
result <- cosinor.analysis(counts$axis1, counts$timestamp)
print(result)
```

`cosinor.antilogistic` *Anti-Logistic (Extended) Cosinor Analysis (Marler et al. 2006)*

Description

Fits the sigmoidally transformed cosine ("anti-logistic" extended cosinor) model of Marler et al. (2006) to the averaged 24-hour activity profile. The extended model relaxes the symmetric shape of the ordinary cosinor by adding two shape parameters: `alpha` sets the relative width of the active versus rest phase, and `beta` sets the steepness of the rest-to-active transitions. It is the same model as `ActCR::ActExtendCosinor` and reproduces its parameter estimates numerically.

Usage

```
cosinor.antilogistic(counts, timestamps, period = 24)
```

Arguments

`counts`	Numeric vector of activity counts (one value per epoch).
`timestamps`	POSIXct vector of timestamps, the same length as `counts`.
`period`	Numeric period of the rhythm in hours (default `24`).

Details

The model fitted to the averaged profile is

$$f(t) = minimum + amplitude \cdot expit(\beta(cos(2\pi(t - acrotime)/T) - \alpha))$$

where $expit(z) = 1/(1+e^{-z})$. This is the ActCR/Marler parameterization, in which `amplitude` is the raw multiplier of the logistic transform (it is *not* renormalized by $expit(\beta(1-\alpha))$). The closely related "normalized" Marler form, $f(t) = min + amp \cdot expit(\beta(cos(\cdot) - \alpha))/expit(\beta(1-\alpha))$, rescales `amplitude` so that the peak equals exactly `minimum + amplitude`; the two forms describe the same curve shape and the (`alpha`, `beta`, `acrotime`) parameters are identical. The normalized peak level is reported as `peak`.

Starting values are taken from an ordinary cosinor fit (`minimum = max(mesor - amp, 0)`, `amplitude = 2 * amp`, `alpha = 0`, `beta = 2`, `acrotime = ordinary acrophase`). The nonlinear least-squares problem is solved with `stats::optim()` using the box-constrained L-BFGS-B method (bounds `lower = c(0, 0, -1, 0, -3)`, `upper = c(Inf, Inf, 1, Inf, 27)`, matching ActCR), so no extra package dependency is required.

Value

A list with class "actiRhythm_cosinor_ext" containing:

minimum Lower asymptote of the fitted curve (rest-phase level).

amplitude Vertical span of the sigmoidal transition (amp); the asymptotic ceiling is minimum + amplitude, with the fitted-curve maximum slightly below it at a finite steepness.

alpha Width-asymmetry parameter in $[-1, 1]$.

beta Steepness parameter (≥ 0).

acrophase Time of peak activity in clock hours (acrotime).

acrotime Alias of acrophase (ActCR naming).

peak Fitted activity level at the acrophase (renormalized Marler peak).

UpMesor Clock time of the rest-to-active (rising) transition, $-\arccos(\alpha)/(2\pi/T) + acrotime$.

DownMesor Clock time of the active-to-rest (falling) transition, $\arccos(\alpha)/(2\pi/T) + acrotime$.

MESOR Midline statistic minimum + amplitude / 2.

F_pseudo Pseudo-F improvement of the extended fit over the ordinary cosinor: $((RSS_{cos} - RSS_{ext})/2)/(RSS_{ext}/(n-5))$.

rss_cosinor Residual sum of squares of the ordinary cosinor.

rss_extended Residual sum of squares of the extended cosinor.

converged Logical; whether the nonlinear fit converged.

period Period used (hours).

n_profile_hours Number of profile hours used in the fit.

On non-convergence or insufficient data all numeric parameters are returned as NA with converged = FALSE.

References

Marler MR, Gehrman P, Martin JL, Ancoli-Israel S (2006). "The sigmoidally transformed cosine curve: a mathematical model for circadian rhythms with symmetric non-sinusoidal shapes." *Statistics in Medicine*, **25**(22), 3893–3904. doi:10.1002/sim.2466.

Di J, Zipunnikov V (2021). *ActCR: Extract Circadian Rhythms Metrics from Actigraphy Data*. R package version 0.2.0, https://CRAN.R-project.org/package=ActCR.

Examples

```
ts <- seq(as.POSIXct("2024-01-01 00:00:00"), by = 60, length.out = 1440 * 3)
hour <- as.numeric(format(ts, "%H")) + as.numeric(format(ts, "%M")) / 60
counts <- 50 + 300 * plogis(4 * (cos((hour - 14) * 2 * pi / 24) - 0.2))
fit <- cosinor.antilogistic(counts, ts)
print(fit)
```

`cosinor.compare` *Compare Cosinor Rhythms Between Two Groups*

Description

Tests whether the rest-activity rhythm differs between two groups. Fits each subject's cosinor with the same engine as `cosinor.analysis`. An Bingham et al. (1982) comparison is a 2-variate Hotelling's T^2 on the cosine and sine coefficients (testing equal amplitude and acrophase, accounting for their correlation), with the MESOR compared by a separate pooled-variance F-test. Auxiliary per-parameter two-sample (Welch) t-tests break the result down for the MESOR, amplitude, and acrophase; the acrophase test is circular-aware, unwrapping the per-subject acrophases about their common circular mean.

Usage

```
cosinor.compare(
  activity,
  timestamps,
  subject,
  group,
  period = 24,
  level = 0.95,
  min_valid_hours = 12
)
```

Arguments

`activity`	Numeric activity vector with all subjects stacked.
`timestamps`	POSIXct timestamps, one per value.
`subject`	Subject identifier, one per value.
`group`	Group identifier (exactly two levels), one per value.
`period`	Rhythm period in hours (default 24).
`level`	Confidence level for the difference CIs (default 0.95).
`min_valid_hours`	Minimum profile hours for a subject to be included.

Value

An object of class `actiRhythm_cosinor_compare`: a `joint` list (the amplitude/acrophase Hotelling T^2 with its F, degrees of freedom and p-value, plus a separate MESOR F-test), a `tests` data frame of auxiliary per-parameter Welch comparisons (each group's estimate, the difference, the t statistic, p-value and CI), and the per-subject coefficients.

References

Bingham C, Arbogast B, Cornelissen Guillaume G, Lee JK, Halberg F (1982). "Inferential statistical methods for estimating and comparing cosinor parameters." *Chronobiologia*, **9**(4), 397–439.

See Also

population.cosinor

Examples

```
set.seed(1); hrs <- 0:23
act <- ts <- subj <- grp <- NULL
for (g in c("A", "B")) for (i in 1:5) {
  acro <- if (g == "A") 8 else 11
  y <- 100 + 40 * cos(2 * pi * (hrs - acro) / 24) + rnorm(24, 0, 4)
  act <- c(act, y); subj <- c(subj, rep(paste0(g, i), 24)); grp <- c(grp, rep(g, 24))
  ts <- c(ts, as.POSIXct("2024-01-01", tz = "UTC") + hrs * 3600)
}
cosinor.compare(act, as.POSIXct(ts, tz = "UTC", origin = "1970-01-01"), subj, grp)
```

---

cosinor.confidence.ellipse

Joint Amplitude-Acrophase Confidence Ellipse (Bingham et al. 1982)

Description

Computes the joint confidence region for the cosinor amplitude and acrophase, following Bingham et al. (1982). The region is an ellipse in the (β_1, β_2) (cosine/sine coefficient) plane. When the ellipse excludes the origin, the population amplitude differs significantly from zero and a rhythm is detected.

Usage

```
cosinor.confidence.ellipse(cosinor_result, level = 0.95, n_points = 200)
```

Arguments

`cosinor_result`	A list returned by `cosinor.analysis()` (class `"actiRhythm_cosinor"`). It must contain `amplitude`, `acrophase` (clock hours), `se_amplitude`, `period` and `n_profile_hours`. If the object additionally carries an explicit `vcov_beta` (2x2 covariance of (β_1, β_2)) and/or `beta1`/`beta2`, those are used directly.
`level`	Confidence level for the region (default `0.95`).
`n_points`	Number of points used to trace the ellipse boundary (default `200`).

Details

Under the cosinor model $Y(t) = M + \beta_1 cos(\omega t) + \beta_2 sin(\omega t)$, the $100(1-\alpha)\%$ joint confidence region for (β_1, β_2) is the set of points b satisfying

$$(b - \hat{b})^\top \Sigma^{-1} (b - \hat{b}) \leq 2F_{2,df,1-\alpha}$$

where Σ is the covariance of the estimated coefficients and df the residual degrees of freedom. The averaged-profile cosinor design used by `cosinor.analysis()` has orthogonal cosine and sine columns, so Σ is (very nearly) diagonal with equal variances σ^2; in that case `se_amplitude` equals the standard error of each coefficient $\sqrt{\mathrm{Var}(\beta)}$, which is how Σ is reconstructed when it is not supplied explicitly. The point estimate is recovered from amplitude and acrophase as $\beta_1 = Acos(\phi)$, $\beta_2 = Asin(\phi)$ with $\phi = acrophase \cdot 2\pi/T$.

Value

A list with class `"actiRhythm_cosinor_ellipse"` containing:

ellipse Data frame with columns `x` (β_1) and `y` (β_2) giving the ellipse boundary vertices.

center Numeric vector `c(beta1, beta2)` of the point estimate.

excludes_origin Logical; `TRUE` when the ellipse excludes `(0, 0)` (i.e. a rhythm is detected at the requested level).

rhythm_detected Alias of `excludes_origin`.

distance_stat Mahalanobis-type statistic of the origin relative to the fitted ellipse.

critical_value Threshold $2F_{2,df,level}$ the statistic is compared against.

level Confidence level used.

When the input is missing required fields or is degenerate, the boundary is returned as `NA` and `excludes_origin = NA`.

References

Bingham C, Arbogast B, Cornelissen Guillaume G, Lee JK, Halberg F (1982). "Inferential statistical methods for estimating and comparing cosinor parameters." *Chronobiologia*, **9**(4), 397–439.

Examples

```
ts <- seq(as.POSIXct("2024-01-01", tz = "UTC"), by = 60, length.out = 1440 * 3)
counts <- 100 + 80 * cos(2 * pi * (as.numeric(format(ts, "%H")) - 14) / 24)
cos <- cosinor.analysis(counts, ts)
cosinor.confidence.ellipse(cos)$rhythm_detected
```

`cosinor.extended` *Multi-Component Cosinor Analysis with Harmonics*

Description

Fits a multi-component cosinor model with the fundamental (24h) and harmonics (12h, 8h, etc.) to capture patterns the single cosinor misses, such as bimodal rhythms.

Usage

```
cosinor.extended(
  counts,
  timestamps,
  harmonics = c(24, 12),
  wear_time = NULL,
  min_valid_hours = 10
)
```

Arguments

counts	Numeric vector of activity counts
timestamps	POSIXct vector of timestamps
harmonics	Vector of periods to include (default: c(24, 12) for 24h + 12h)
wear_time	Optional logical vector indicating valid wear time
min_valid_hours	Numeric. Valid-day criterion (GGIR includedaycrit): minimum wear hours for a day to count; default 10, 0/NULL to disable.

Details

The multi-component model is:

`Y(t) = M + sum_k[A_k * cos(2*pi*t/T_k - phi_k)]`

Where T_k are the periods (24h, 12h, 8h, etc.)

Use it for:

- Bimodal patterns (morning + evening peaks), captured by the 12h harmonic
- Complex daily routines with multiple activity bouts
- Shift workers or irregular schedules

Value

A list with class "actiRhythm_cosinor_extended" containing:

mesor Rhythm-adjusted mean

components Data frame with amplitude, acrophase, acrophase_time for each harmonic

dominant_period Period with largest amplitude

dominant_acrophase Acrophase of dominant component

r_squared Model R-squared (should be higher than single-component)

r_squared_improvement Improvement over single 24h cosinor

References

Cornelissen G (2014). "Cosinor-based rhythmometry." *Theoretical Biology and Medical Modelling*, **11**, 16. doi:10.1186/174246821116.

Refinetti R, Cornelissen G, Halberg F (2007). "Procedures for numerical analysis of circadian rhythms." *Biological Rhythm Research*, **38**(4), 275–325. doi:10.1080/09291010600903692.

Examples

```
counts <- agd.counts(read.agd(example_agd()))
result <- cosinor.extended(counts$axis1, counts$timestamp, harmonics = c(24, 12, 8))
print(result)
```

`cosinor.multicomponent`

Multicomponent Cosinor with Model Selection

Description

Fits the multi-component (harmonic) cosinor of Cornelissen (2014) with one to several harmonics of the fundamental period and picks the number of harmonics by an information criterion (AIC or BIC, a package choice), so it captures a bimodal or asymmetric daily shape without your choosing the order by hand. The single cosinor is the one-harmonic special case.

Usage

```
cosinor.multicomponent(
  counts,
  timestamps,
  period = 24,
  max_harmonics = 3,
  criterion = c("AIC", "BIC")
)
```

Arguments

`counts`	Numeric activity vector.
`timestamps`	POSIXct timestamps, one per value.
`period`	Fundamental period in hours (default 24).
`max_harmonics`	Largest number of harmonics to consider (default 3).
`criterion`	`"AIC"` (default) or `"BIC"` for model selection.

Value

An object of class `actiRhythm_multicosinor`: the selected number of harmonics, the per-harmonic amplitude and acrophase, the MESOR, R-squared, and the full model-comparison table. Never errors.

References

Cornelissen G (2014). "Cosinor-based rhythmometry." *Theoretical Biology and Medical Modelling*, **11**, 16. doi:10.1186/174246821116.

Examples

```
ts <- seq(as.POSIXct("2024-01-01", tz = "UTC"), by = 60, length.out = 4 * 1440)
th <- as.numeric(difftime(ts, ts[1], units = "hours"))
y  <- 100 + 40 * cos(2 * pi * th / 24) + 20 * cos(2 * pi * 2 * th / 24)
cosinor.multicomponent(y, ts)
```

`counts.from.data.frame`

Activity Counts from a Data Frame

Description

A device-neutral entry point: pull a count column (and optionally a time column) out of any data frame and return a tidy `timestamp`/`counts` frame ready for the analysis functions. Use it for non-ActiGraph counts or any pre-extracted series.

Usage

```
counts.from.data.frame(
  df,
  count_col = "axis1",
  time_col = NULL,
  epoch_length = 60,
  tz = "UTC",
  start = "1970-01-01"
)
```

Arguments

`df`	A data frame.
`count_col`	Name or index of the count column (default `"axis1"`).
`time_col`	Name or index of the timestamp column; if `NULL`, timestamps are synthesized from `start` by `epoch_length` (with a warning).
`epoch_length`	Epoch length in seconds for synthesized timestamps (default 60).
`tz`	Time zone (default `"UTC"`).
`start`	Start time for synthesized timestamps (default `"1970-01-01"`).

Value

A data frame with `timestamp` (POSIXct) and `counts`.

See Also

`read.actigraph.csv`

Examples

```
df <- data.frame(activity = c(0, 50, 300), clock = c("2024-01-01 00:00:00",
  "2024-01-01 00:01:00", "2024-01-01 00:02:00"))
counts.from.data.frame(df, count_col = "activity", time_col = "clock")
```

`cpp_available` *Check C++ Availability*

Description

Check C++ Availability

Usage

```
cpp_available()
```

Value

Logical

`curve.registration` *Curve Registration of Daily Activity Profiles*

Description

Aligns each day's 24-hour activity profile on its active-phase landmark (the M10 centre), separating the horizontal phase variation (how the timing shifts day to day) from the vertical amplitude variation (the registered mean profile, sharper than the plain average because phase jitter no longer blurs it). The per-day landmark times are a scale-invariant phase marker (the M10-window centre), unchanged by any rescaling of the counts (Ramsay and Silverman 2005).

Usage

```
curve.registration(counts, timestamps, n_grid = 144L, period = 24)
```

Arguments

`counts`	Numeric activity vector.
`timestamps`	POSIXct timestamps, one per value.
`n_grid`	Bins per day for the profile (default 144 = 10-minute bins).
`period`	Period in hours (default 24).

Value

An object of class `actiRhythm_registration`: a per-day landmark table (L5 and M10 centre hours), the circular-mean landmarks, the phase variability (circular SD of the M10 landmark), and the registered mean profile. Never errors.

References

Ramsay JO, Silverman BW (2005). *Functional Data Analysis*, 2nd edition. Springer. doi:10.1007/b98888.

Van Someren EJW, Swaab DF, Colenda CC, Cohen W, McCall WV, Rosenquist PB (1999). "Bright light therapy: improved sensitivity to its effects on rest-activity rhythms in Alzheimer patients by application of nonparametric methods." *Chronobiology International*, **16**(4), 505–518. doi:10.3109/07420529908998724.

Examples

```
ts <- seq(as.POSIXct("2024-01-01", tz = "UTC"), by = 60, length.out = 5 * 1440)
h  <- as.numeric(format(ts, "%H"))
curve.registration(ifelse(h >= 23 | h < 7, 5, 300), ts)
```

`detect.nonwear.choi` *Choi (2011) Non-Wear Detection*

Description

Classifies each epoch as wear or non-wear with the Choi et al. (2011) algorithm: a run of consecutive zero-count epochs of at least `frame` minutes is non-wear, tolerating a short nonzero spike only when it is flanked by a fully zero window of `stream` minutes both before and after. The returned mask can be passed as the `wear_time` argument to the rest-activity and sleep functions.

Usage

```
detect.nonwear.choi(
  counts,
  epoch_length = 60,
  frame = 90,
  spike_tolerance = 2,
  stream = 30
)
```

Arguments

`counts`	Numeric activity vector (vertical axis), minute epochs assumed.
`epoch_length`	Epoch length in seconds (default 60). Window lengths are given in minutes and scaled to epochs by this value.

`frame` Minimum non-wear window in minutes (default 90).

`spike_tolerance` Maximum tolerated nonzero spike in minutes (default 2).

`stream` Flanking all-zero window required around a tolerated spike, in minutes (default 30).

Value

A logical vector, one per epoch: `TRUE` = wear, `FALSE` = non-wear. Never errors.

References

Choi L, Liu Z, Matthews CE, Buchowski MS (2011). "Validation of accelerometer wear and non-wear time classification algorithm." *Medicine & Science in Sports & Exercise*, **43**(2), 357–364. doi:10.1249/MSS.0b013e3181ed61a3.

See Also

`detect.nonwear.troiano`

Examples

```
ts <- seq(as.POSIXct("2024-01-01", tz = "UTC"), by = 60, length.out = 600)
counts <- c(rep(200, 200), rep(0, 200), rep(200, 200))   # 200-min non-wear gap
table(detect.nonwear.choi(counts))
```

`detect.nonwear.raw` *Non-Wear Detection from Raw Acceleration*

Description

Flags non-wear time from raw acceleration by the van Hees et al. (2013) / GGIR standard-deviation-and-range rule: a block is non-wear when, over a window centred on it (default 60 minutes), at least two of the three axes have both a standard deviation below `sd_crit` and a value range below `range_crit`. A stationary, taken-off device reads as non-wear. This lets the z-angle sleep detector tell device-off periods from real sleep (a still arm that keeps micro-movement). Pass the result as the `wear` argument of `rest.spt`.

Usage

```
detect.nonwear.raw(
  x,
  device = "auto",
  epoch = 5,
  block = 300,
  window = 3600,
  sd_crit = 0.013,
```

```
  range_crit = 0.05,
  tz = "UTC"
)
```

Arguments

x	A path to a raw file or a raw data frame (see raw.metrics).
device	Device brand or "auto" (file input only).
epoch	Output epoch length in seconds for the returned mask (default 5, matching rest.spt).
block	Internal classification block length in seconds (default 300; GGIR steps the 60-minute window in 15-minute blocks, so boundaries can differ by a few minutes from GGIR's grid).
window	Window in seconds over which the SD and range are taken (default 3600).
sd_crit	Per-axis SD threshold in g (default 0.013).
range_crit	Per-axis range threshold in g (default 0.050).
tz	Time zone (default "UTC").

Value

A logical vector, one per epoch: TRUE = wear, FALSE = non-wear. Never errors.

References

van Hees VT, Gorzelniak L, Dean Leon EC, Eder M, Pias M, Taherian S, Ekelund U, Renstrom F, Franks PW, Horsch A, Brage S (2013). "Separating movement and gravity components in an acceleration signal and implications for the assessment of human daily physical activity." *PLoS ONE*, **8**(4), e61691. doi:10.1371/journal.pone.0061691.

See Also

rest.spt, raw.metrics

Examples

```
raw  <- example_raw(days = 2, device_off = 1)   # two worn days + one device-off day
mean(detect.nonwear.raw(raw, epoch = 60))       # fraction of epochs worn
```

`detect.nonwear.troiano` *Troiano (2008) Non-Wear Detection*

Description

Classifies each epoch as wear or non-wear with the Troiano et al. (2008, NHANES) algorithm: a run of at least `frame` minutes of zero counts is non-wear, tolerating up to `spike_tolerance` consecutive nonzero minutes only if every one of them is at or below `stoplevel` counts. Unlike Choi it has no flanking-window requirement but applies a count ceiling on spikes.

Usage

```
detect.nonwear.troiano(
  counts,
  epoch_length = 60,
  frame = 60,
  spike_tolerance = 2,
  stoplevel = 100
)
```

Arguments

`counts`	Numeric activity vector (vertical axis), minute epochs assumed.
`epoch_length`	Epoch length in seconds (default 60).
`frame`	Minimum non-wear window in minutes (default 60).
`spike_tolerance`	Maximum tolerated nonzero spike in minutes (default 2).
`stoplevel`	Count above which a spike ends the non-wear bout (default 100).

Value

A logical vector, one per epoch: `TRUE` = wear, `FALSE` = non-wear. Never errors.

References

Troiano RP, Berrigan D, Dodd KW, Masse LC, Tilert T, McDowell M (2008). "Physical activity in the United States measured by accelerometer." *Medicine & Science in Sports & Exercise*, **40**(1), 181–188. doi:10.1249/mss.0b013e31815a51b3.

See Also

`detect.nonwear.choi`

Examples

```
ts <- seq(as.POSIXct("2024-01-01", tz = "UTC"), by = 60, length.out = 400)
counts <- c(rep(150, 100), rep(0, 200), rep(150, 100))
table(detect.nonwear.troiano(counts))
```

---

dichotomy.index *Dichotomy Index (I < O)*

Description

The fraction of in-rest activity counts that fall below the median of the out-of-rest (active) counts, a rest/active separation index used in cancer chronobiology (Mormont et al. 2000). A high I<O means rest is quiet relative to the active day, marking a well-separated rhythm.

Usage

```
dichotomy.index(counts, rest)
```

Arguments

`counts`	Numeric activity vector.
`rest`	A logical vector (TRUE = rest / in-bed), or a character vector of states where `"R"`/`"S"`/`"sleep"` mark rest. Same length as `counts`.

Value

An object of class `actiRhythm_dichotomy`: the index `IO` (percent), the active-period median, and epoch counts. Never errors.

References

Mormont MC, Waterhouse J, Bleuzen P, Giacchetti S, Jami A, Bogdan A, Lellouch J, Misset JL, Touitou Y, Levi F (2000). "Marked 24-h rest/activity rhythms are associated with better quality of life, better response, and longer survival in patients with metastatic colorectal cancer and good performance status." *Clinical Cancer Research*, **6**(8), 3038–3045.

Examples

```
ts <- seq(as.POSIXct("2024-01-01", tz = "UTC"), by = 60, length.out = 2 * 1440)
h  <- as.numeric(format(ts, "%H"))
counts <- ifelse(h >= 23 | h < 7, 5, 300)
dichotomy.index(counts, rest = h >= 23 | h < 7)
```

example_agd *Get Path to Example AGD Files*

Description

Get Path to Example AGD Files

Usage

```
example_agd(file = 1)
```

Arguments

file	Character. Name of the example file or "list" to see available files.

Value

Character. Full path to the example AGD file.

Examples

```
example_agd()
example_agd("list")
agd_path <- example_agd(1)
```

example_raw *Synthetic Raw Acceleration Recording*

Description

Generates a deterministic synthetic triaxial raw acceleration recording (in g) with a day/night posture cycle, daytime movement and posture changes, still (sleeping) nights, and a slight built-in miscalibration. It is the file-free stand-in for a real raw file in the examples: feed it to raw.metrics, circadian.raw, or the z-angle sleep pipeline. Raw acceleration is far too large to ship as data, so it is generated on demand rather than bundled.

Usage

```
example_raw(days = 2, fs = 30, device_off = 0)
```

Arguments

days	Recording length in days (default 2, giving two nights).
fs	Sample rate in Hz (default 30).
device_off	Days of a still, taken-off device to append at the end (default 0). Use it to exercise detect.nonwear.raw and the non-wear gate of rest.spt.

Value

A data frame with a POSIXct `time` column and x/y/ z acceleration in g, with the sample rate in the `"fs"` attribute.

See Also

`raw.metrics`, `circadian.raw`, `rest.spt`

Examples

```
raw <- example_raw(days = 1)
str(raw)
```

`fractal.dfa` *Detrended Fluctuation Analysis (DFA)*

Description

Computes the scaling exponent alpha of an activity time series using Detrended Fluctuation Analysis (Peng et al., 1994). DFA quantifies long-range temporal correlations: alpha approximately 0.5 indicates uncorrelated (white) noise, alpha approximately 1.0 indicates 1/f (pink) noise, and alpha approximately 1.5 indicates Brownian (random-walk / brown) noise. Healthy human activity fluctuations typically show alpha in the 0.9 to 1.0 range, with reductions reported in aging and Alzheimer's disease (Hu et al., 2009); those reference values were obtained with quadratic DFA-2 (`detrend_order = 2`), which can differ from the default linear DFA-1 on trend-carrying data.

Usage

```
fractal.dfa(
  x,
  scale_min = 4,
  scale_max = NULL,
  breakpoint_min = 90,
  detrend_order = 1L
)
```

Arguments

`x`	Numeric vector of activity counts (minute-level recommended). Internally analyzed on the longest continuous non-NA segment.
`scale_min`	Integer. Smallest window size (box length) in samples. Must be >= 4 so that a line can be detrended with residual degrees of freedom. Default 4.
`scale_max`	Integer or NULL. Largest window size in samples. If NULL (default) it is set to floor(N / 4) where N is the length of the analyzed segment, ensuring at least four windows at the largest scale. The top scales then have only about four windows and are somewhat undersampled; Hu et al. (2001) suggest a maximum nearer N / 10 for well-sampled fluctuations.

`breakpoint_min` Numeric. Window-size boundary (in samples / minutes for minute-level data) separating the short-timescale exponent `alpha1` (scales < breakpoint_min) from the long-timescale exponent `alpha2` (scales >= breakpoint_min). Default 90.

`detrend_order` Integer order of the within-window polynomial detrend: 1 (default) is linear DFA-1 (Peng et al. 1994); 2 is quadratic DFA-2, the convention Hu et al. (2009) used for activity data.

Details

Algorithm (integrated, non-overlapping DFA, polynomial detrend of order `detrend_order`):

1. Extract the longest continuous non-NA segment of `x`.
2. Integrate the mean-centred signal: `y = cumsum(x - mean(x))`.
3. For each window size n (log-spaced from `scale_min` to `scale_max`), split y into floor(N / n) non-overlapping windows of length n, fit and remove a least-squares polynomial of order `detrend_order` within each window, and pool the residuals. The fluctuation is `F(n) = sqrt(mean(residuals^2))` over all pooled residuals.
4. The scaling exponent is the slope of `log10(F(n))` regressed on `log10(n)`.

Window sizes are unique integers chosen on a base-10 log grid, which gives approximately even spacing in the log-log fit and matches the convention used by reference implementations such as **nonlinearTseries**.

Value

A list with class `"actiRhythm_dfa"` containing:

alpha Overall scaling exponent: slope of `lm(log10(F) ~ log10(n))` across all scales.

alpha1 Short-timescale exponent (scales < `breakpoint_min`). NA if fewer than two qualifying scales.

alpha2 Long-timescale exponent (scales >= `breakpoint_min`). NA if fewer than two qualifying scales.

scales Integer vector of window sizes n that were used.

fluctuations Numeric vector of fluctuation magnitudes F(n) corresponding to `scales`.

n_used Length of the analyzed (longest non-NA) segment.

breakpoint_min The breakpoint value used.

On an unusable series (too short, all NA, or constant) the numeric scaling outputs are returned as NA with the same structure (never an error).

See Also

`multiscale.entropy`, `circadian.rhythm`

Examples

```
set.seed(1)
# White noise -> alpha near 0.5
fractal.dfa(rnorm(10000))$alpha
# Brown noise -> alpha near 1.5
fractal.dfa(cumsum(rnorm(10000)))$alpha
```

---

geneactiv.counts *Activity Counts from a Raw GENEActiv .bin File*

Description

Reads a raw GENEActiv (`.bin`) accelerometer file and converts it to ActiGraph-equivalent activity counts via the agcounts implementation of the ActiGraph count algorithm (Neishabouri 2022). Requires the **GGIRread** and **agcounts** packages.

Usage

```
geneactiv.counts(path, epoch = 60, lfe = FALSE, tz = "UTC")
```

Arguments

`path`	Path to a `.bin` file.
`epoch`	Epoch length in seconds (default 60).
`lfe`	Use the low-frequency extension filter (default `FALSE`).
`tz`	Time zone for the timestamps (default `"UTC"`).

Value

Data frame with `time`, `axis1`, `axis2`, `axis3` and `vm`, one row per epoch.

Cross-brand counts

These are an *approximation* of ActiGraph counts, not native ActiGraph output, and GENEActiv-to-count conversion is *not* empirically validated (only theoretically motivated by the shared filter). GENEActiv records at about 85.7 Hz, which is resampled to 30 Hz before the count filter. Appropriate for the relative and normalized circadian metrics here (IS, IV, RA, L5, M10, SRI); do *not* apply ActiGraph cut-points or compare absolute counts across brands.

References

Neishabouri A, Nguyen J, Samuelsson J, Guthrie T, Biggs M, Wyatt J, Cross D, Karas M, Migueles JH, Khan S, Guo CC (2022). "Quantification of acceleration as activity counts in ActiGraph wearable." *Scientific Reports*, **12**, 11958. doi:10.1038/s4159802216003x.

Brond JC, Andersen LB, Arvidsson D (2017). "Generating ActiGraph counts from raw acceleration recorded by an alternative monitor." *Medicine & Science in Sports & Exercise*, **49**(11), 2351–2360. doi:10.1249/MSS.0000000000001344.

See Also

axivity.counts, gt3x.counts, read.raw

---

gt3x.counts *Activity Counts from a Raw .gt3x File*

Description

Compute activity counts from a raw ActiGraph .gt3x accelerometer file using the agcounts implementation of the ActiGraph count algorithm (Neishabouri 2022). Requires the **agcounts** and **read.gt3x** packages.

Usage

```
gt3x.counts(path, epoch = 60, lfe = FALSE, tz = "UTC")
```

Arguments

path	Path to a .gt3x file.
epoch	Epoch length in seconds (default 60).
lfe	Use the low-frequency extension filter (default FALSE).
tz	Time zone for the timestamps (default "UTC").

Value

Data frame with time, axis1, axis2, axis3 and vm, one row per epoch.

References

Neishabouri A, Nguyen J, Samuelsson J, Guthrie T, Biggs M, Wyatt J, Cross D, Karas M, Migueles JH, Khan S, Guo CC (2022). "Quantification of acceleration as activity counts in ActiGraph wearable." *Scientific Reports*, **12**, 11958. doi:10.1038/s41598022160003x.

See Also

axivity.counts, geneactiv.counts, read.raw

gt3x.to.agd *Convert a Raw .gt3x File to an .agd File*

Description

Computes counts with `gt3x.counts` and writes them to a small `.agd`, carrying device/subject metadata from the `.gt3x` header. Requires **agcounts**.

Usage

```
gt3x.to.agd(gt3x_path, agd_path = NULL, epoch = 60, lfe = FALSE, tz = "UTC")
```

Arguments

`gt3x_path`	Path to the `.gt3x` file.
`agd_path`	Output `.agd` path; defaults next to the input.
`epoch`	Epoch length in seconds (default 60).
`lfe`	Use the low-frequency extension filter (default `FALSE`).
`tz`	Time zone (default `"UTC"`).

Value

The output `.agd` path.

has.inclinometer *Check if AGD Data Has Inclinometer Information*

Description

Checks whether an AGD file contains inclinometer data, which can be used to enhance sleep detection algorithms.

Usage

```
has.inclinometer(agd_data)
```

Arguments

`agd_data`	List returned from read.agd()

Value

Logical. TRUE if inclinometer data is available

`hilbert.huang` *Hilbert-Huang Instantaneous Phase and Frequency*

Description

Computes the instantaneous amplitude, phase, and period of a circadian intrinsic mode function via its analytic signal (Huang et al. 1998). Where the cosinor gives one acrophase for the whole recording, the instantaneous phase tracks the rhythm cycle by cycle, and the spread of the instantaneous period measures how stationary the circadian band is.

Usage

```
hilbert.huang(x, timestamps = NULL, imf = NULL, epoch_length = 60)
```

Arguments

`x`	A `actiRhythm_emd` object, or a numeric activity vector (then `timestamps` is required).
`timestamps`	POSIXct timestamps (required when `x` is numeric).
`imf`	Which IMF to analyse; default is the circadian IMF of the EMD object.
`epoch_length`	Epoch length in seconds (default 60; used when `x` is numeric).

Value

An object of class `actiRhythm_hht`: the instantaneous amplitude, phase, and period series, plus the mean instantaneous period and its SD, mean amplitude and its CV, and the fraction of time the period stays in 20-28 h. Never errors.

References

Huang NE, Shen Z, Long SR, Wu MC, Shih HH, Zheng Q, Yen NC, Tung CC, Liu HH (1998). "The empirical mode decomposition and the Hilbert spectrum for nonlinear and non-stationary time series analysis." *Proceedings of the Royal Society A*, **454**(1971), 903–995. doi:10.1098/rspa.1998.0193.

See Also

`circadian.emd`

Examples

```
ts <- seq(as.POSIXct("2024-01-01", tz = "UTC"), by = 600, length.out = 6 * 144)
th <- as.numeric(difftime(ts, ts[1], units = "hours"))
e <- circadian.emd(100 + 60 * cos(2 * pi * th / 24), ts, epoch_length = 600)
hilbert.huang(e)
```

`intradaily.variability.multiscale` *Multiscale Intradaily Variability (IVm)*

Description

Intradaily variability computed across a set of bin sizes and averaged, the counterpart to multiscale interdaily stability (`circadian.is.multiscale`). Fragmentation that is invisible at the hourly scale often shows at finer bins, so the averaged IVm varies less across recordings than the single hourly IV (Goncalves et al. 2014).

Usage

```
intradaily.variability.multiscale(counts, timestamps, bin_minutes = 1:60)
```

Arguments

`counts`	Numeric activity vector.
`timestamps`	POSIXct timestamps, one per value.
`bin_minutes`	Bin sizes in minutes (default 1 to 60, per Goncalves et al. 2014).

Value

An object of class `actiRhythm_ivm`: a per-bin `IV` table and the averaged `IVm`. Never errors; returns `NA` on insufficient data.

References

Goncalves BSB, Cavalcanti PRA, Tavares GR, Campos TF, Araujo JF (2014). "Nonparametric methods in actigraphy: an update." *Sleep Science*, **7**(3), 158–164. doi:10.1016/j.slsci.2014.09.013.

See Also

`circadian.is.multiscale`

Examples

```
ts <- seq(as.POSIXct("2024-01-01", tz = "UTC"), by = 60, length.out = 3 * 1440)
h  <- as.numeric(format(ts, "%H"))
intradaily.variability.multiscale(ifelse(h >= 23 | h < 7, 5, 300), ts)
```

IS_cpp *Interdaily Stability (IS)*

Description

Interdaily Stability (IS)

Usage

```
IS_cpp(hourly_data, hours_per_day = 24L)
```

Arguments

hourly_data	Numeric vector of hourly activity
hours_per_day	Hours per day (default: 24)

Value

IS value (0-1, higher = more stable)

IV_cpp *Intradaily Variability (IV)*

Description

Intradaily Variability (IV)

Usage

```
IV_cpp(hourly_data)
```

Arguments

hourly_data	Numeric vector of hourly activity

Value

IV value (lower = less fragmented)

L5M10_cpp *Calculate L5/M10 Circadian Metrics*

Description

Finds least active 5 hours (L5) and most active 10 hours (M10) using van Someren (1999) average-profile method.

Usage

```
L5M10_cpp(minute_data, window_L5 = 300L, window_M10 = 600L, start_minute = 0L)
```

Arguments

minute_data	Numeric vector of minute-level activity
window_L5	L5 window in minutes (default: 300)
window_M10	M10 window in minutes (default: 600)
start_minute	Start minute of day (0-1439)

Value

List with L5_value, L5_onset, L5_onset_hours, M10_value, M10_onset, M10_onset_hours, RA

lids *Locomotor Inactivity During Sleep (LIDS)*

Description

Quantifies the ultradian (sleep-cycle) oscillation of inactivity during sleep, following Winnebeck et al. (2018). Within each sleep period the activity is summed into 10-minute bins, transformed to LIDS (`100 / (activity + 1)`), smoothed with a centered 30-minute moving average, and fit with an ultradian cosine over a scan of candidate periods; the best period is the interior peak that maximises the Munich Rhythmicity Index (MRI).

Usage

```
lids(
  counts,
  timestamps,
  sleep_periods,
  epoch_length = 60,
  smooth_minutes = 30,
  period_min = 30,
  period_max = 180,
  period_step = 5
)
```

Arguments

`counts`	Numeric activity vector.
`timestamps`	POSIXct (or parseable) timestamps, one per count.
`sleep_periods`	Data frame with `in_bed_time` and `out_bed_time` (same schema as `social.jet.lag`); one LIDS fit per period.
`epoch_length`	Epoch length in seconds (default 60).
`smooth_minutes`	Centered moving-average window in minutes (default 30).
`period_min`, `period_max`, `period_step`	Period scan grid in minutes (defaults 30, 180, 5).

Value

An object of class `actiRhythm_lids`: a per-period data frame (period in minutes, MRI, amplitude, Pearson r, ...) and the mean period and MRI across periods.

References

Winnebeck EC, Fischer D, Leise T, Roenneberg T (2018). "Dynamics and ultradian structure of human sleep in real life." *Current Biology*, **28**(1), 49–59. doi:10.1016/j.cub.2017.11.063.

`mfdfa` *Multifractal Detrended Fluctuation Analysis (MF-DFA)*

Description

Generalizes detrended fluctuation analysis (`fractal.dfa`) to a spectrum of moment orders q (Kantelhardt et al. 2002). A flat generalized Hurst exponent h(q) indicates a monofractal signal; a decreasing h(q) and a wide multifractal spectrum indicate multifractality. h(2) equals the standard DFA scaling exponent.

Usage

```
mfdfa(
  x,
  scale_min = 8L,
  scale_max = NULL,
  q_values = seq(-5, 5, by = 0.5),
  both_ends = TRUE,
  detrend_order = 1L
)
```

Arguments

x	Numeric series (e.g. activity counts); the longest gap-free run is analysed.
scale_min	Smallest window size in samples (default 8).
scale_max	Largest window size (default floor(N/4)).
q_values	Moment orders to evaluate (default seq(-5, 5, 0.5)).
both_ends	If TRUE (default) windows are taken from both ends so the tail of the profile is not discarded; FALSE reproduces the start-only convention of fractal.dfa.
detrend_order	Integer order of the within-window polynomial detrend (default 1, linear MF-DFA1; 2 gives MF-DFA2, matching DFA-2).

Value

An object of class actiRhythm_mfdfa: a list with q_values, h_q (generalized Hurst exponent), tau_q (mass exponent), alpha/f_alpha (the multifractal spectrum), alpha_dfa (= h(2)) and the spectrum width. Returns an NA structure on insufficient data; never errors.

References

Kantelhardt JW, Zschiegner SA, Koscielny-Bunde E, Havlin S, Bunde A, Stanley HE (2002). "Multifractal detrended fluctuation analysis of nonstationary time series." *Physica A: Statistical Mechanics and its Applications*, **316**(1-4), 87–114. doi:10.1016/S03784371(02)013833.

See Also

fractal.dfa

Examples

```
set.seed(1)
mfdfa(stats::rnorm(4096))$alpha_dfa   # near 0.5 for white noise
```

---

multiscale.entropy *Multiscale Sample Entropy (MSE)*

Description

Computes Multiscale Sample Entropy (Costa et al., 2002, 2005), which applies Sample Entropy (Richman & Moorman, 2000) to coarse-grained versions of an activity time series across a range of temporal scales. MSE distinguishes genuinely complex signals (whose entropy is sustained or increases across scales, e.g. 1/f-like physiological signals) from uncorrelated random signals (whose entropy falls monotonically with scale).

Usage

```
multiscale.entropy(x, scales = 1:20, m = 2, r = 0.15)
```

Arguments

x	Numeric vector of activity counts (minute-level recommended). Internally analyzed on the longest continuous non-NA segment.
scales	Integer vector of coarse-graining scale factors tau. Default 1:20.
m	Integer embedding dimension (template length) for Sample Entropy. Default 2.
r	Numeric tolerance as a fraction of the standard deviation of the ORIGINAL (scale-1) series. The Chebyshev matching radius used at every scale is r * sd(x_original), following the standard MSE convention of holding r fixed in absolute units across scales. Default 0.15.

Details

For each scale tau the series is coarse-grained into non-overlapping means of length tau, then Sample Entropy is computed on the coarse-grained series with embedding dimension m and the FIXED absolute tolerance r * sd(x_original). Sample Entropy is the negative natural log of the conditional probability that two sub-sequences matching for m points (within the tolerance, Chebyshev distance, self-matches excluded) also match for m + 1 points.

Value

A list with class "actiRhythm_mse" containing:

mse Numeric vector of SampEn values, one per requested scale (NA where the coarse-grained series is too short or yields no matches).

scales Integer vector of the scale factors used.

area Sum of mse over scales (complexity index), sum(mse, na.rm = TRUE).

slope Slope of lm(mse ~ scales) over the non-NA points (negative => entropy declines with scale, typical of noise).

r_absolute The absolute tolerance r * sd(x_original) used.

n_used Length of the analyzed (longest non-NA) segment.

On an unusable series the entropy vector is all NA with the same structure (never an error).

See Also

fractal.dfa, circadian.rhythm

Examples

```
set.seed(1)
# White noise: entropy decreases with scale
multiscale.entropy(rnorm(2000))$mse
# 1/f-like (random walk): flatter / sustained entropy
multiscale.entropy(cumsum(rnorm(2000)))$mse
```

period.ci *Confidence Interval for the Endogenous Circadian Period*

Description

Attaches a bootstrap confidence interval (and standard error) to the Lomb-Scargle period estimate from circadian.period. Because activity is strongly autocorrelated, an i.i.d. resample would destroy the rhythm and give an invalid (too-narrow) interval, so a *circular moving-block* bootstrap of the cosinor residuals is used: the rhythm fitted at the point estimate is held fixed while blocks of residuals are resampled, the periodogram peak is re-located on each replicate (with sub-grid parabolic refinement), and the interval is taken from the quantiles of the replicate periods.

Usage

```
period.ci(
  counts,
  timestamps,
  from = 18,
  to = 30,
  ofac = 4,
  n_boot = 200,
  block_hours = 24,
  level = 0.95,
  seed = NULL
)
```

Arguments

counts	Numeric activity vector; NA are dropped with their times.
timestamps	POSIXct (or numeric-coercible) timestamps, same length.
from, to	Period search window in hours (default 18, 30).
ofac	Integer oversampling factor for the period grid (default 4).
n_boot	Number of bootstrap replicates (default 200).
block_hours	Moving-block length in hours (default 24).
level	Confidence level (default 0.95).
seed	Optional integer seed passed to set.seed for reproducible bootstrap draws.

Value

An object of class actiRhythm_period_ci: a list with tau (the point estimate, hours), ci_lower/ci_upper, se, the level, the number of valid replicates, and tau_boot (the vector of bootstrap replicate periods).

References

Kunsch HR (1989). "The jackknife and the bootstrap for general stationary observations." *The Annals of Statistics*, **17**(3), 1217–1241. doi:10.1214/aos/1176347265.

Politis DN, Romano JP (1992). "A circular block-resampling procedure for stationary data." In LePage R, Billard L (eds.), *Exploring the Limits of Bootstrap*, 263–270. Wiley, New York.

See Also

`circadian.period`

Examples

```
t_hours <- seq(0, 5 * 24 - 1/60, by = 1/60)
ts <- as.POSIXct("2024-01-01", tz = "UTC") + t_hours * 3600
counts <- 100 + 80 * cos(2 * pi * (t_hours - 8) / 24) + rnorm(length(t_hours), 0, 20)
period.ci(counts, ts, n_boot = 50, seed = 1)
```

`phase.concentration` *Phase Concentration Tests*

Description

Tests whether a set of daily phase markers (acrophases, onsets, L5/M10 times) are concentrated rather than scattered around the clock. Reports the mean resultant vector, the Rayleigh test of uniformity (Fisher 1993), and the Hermans-Rasson test (Landler et al. 2019), which catches multimodal clustering that Rayleigh misses.

Usage

```
phase.concentration(times_h, period = 24, n_perm = 2000)
```

Arguments

`times_h`	Numeric vector of clock times (hours, 0-24), one per day.
`period`	Period the times wrap on, in hours (default 24).
`n_perm`	Permutations for the Hermans-Rasson p-value (default 2000, a speed tradeoff; Landler et al. 2019 use 9999 for a finer minimum p-value).

Value

An object of class `actiRhythm_phasetest`: mean direction, mean resultant length R, and the Rayleigh and Hermans-Rasson statistics and p-values. Never errors.

References

Fisher NI (1993). *Statistical Analysis of Circular Data*. Cambridge University Press. doi:10.1017/CBO9780511564345.

Landler L, Ruxton GD, Malkemper EP (2019). "The Hermans-Rasson test as a powerful alternative to the Rayleigh test for circular statistics in biology." *BMC Ecology*, **19**, 30. doi:10.1186/s12898-01902468.

Examples

```
set.seed(1)
onsets <- 23 + stats::rnorm(10, 0, 0.5)      # tightly clustered near 23:00
phase.concentration(onsets %% 24)
```

`plot.actiRhythm_circadian`

Plot Circadian Rhythm Profile

Description

Plots the circadian rhythm analysis.

Usage

```
## S3 method for class 'actiRhythm_circadian'
plot(x, type = "profile", ...)
```

Arguments

`x`	actiRhythm_circadian object from circadian.rhythm()
`type`	Type of plot: "profile" (default), "daily", or "all"
`...`	Additional arguments passed to plotting functions

Value

ggplot object (or list of ggplot objects if type="all")

plot_actogram *Double-Plotted Actogram*

Description

Draws a classic double-plotted actogram: one row per calendar day, each row showing 48 hours (the day itself on the left, the following day on the right) so circadian phase can be traced down the diagonal. Activity is rendered as a per-minute raster (darker = more active). Non-wear (via `wear_time`) and missing time are left blank, and any skipped calendar days appear as empty rows.

Usage

```
plot_actogram(
  counts,
  timestamps,
  epoch_length = NULL,
  wear_time = NULL,
  double_plot = TRUE,
  L5_onset = NULL,
  M10_onset = NULL,
  sleep_mask = NULL,
  scale = c("linear", "sqrt")
)
```

Arguments

`counts`	Numeric vector of activity counts on a regular epoch grid.
`timestamps`	A `POSIXct` vector the same length as `counts`.
`epoch_length`	Epoch length in seconds. If `NULL` (default) it is inferred from the median spacing of `timestamps`.
`wear_time`	Optional logical vector (`TRUE` = worn) the same length as `counts`; non-wear epochs are blanked.
`double_plot`	Logical; draw the 48-hour double plot (default `TRUE`) or a single 24-hour plot.
`L5_onset, M10_onset`	Optional L5 / M10 onset to overlay as a dashed vertical phase line. Accepts a decimal hour, an `"HH:MM"` string, or a `POSIXct`.
`sleep_mask`	Optional logical/character vector (`TRUE`/"S" = asleep) the same length as `counts`; sleep is shaded over the raster.
`scale`	Fill scaling: `"linear"` (default) or `"sqrt"` to compress a heavy-tailed activity range.

Value

A `ggplot` object. Never errors; returns an annotated empty plot on insufficient data.

plot_changepoints *Sleep / Wake Change-Point Track*

Description

Plots the activity series with the per-night sleep-onset and wake-onset change points from `sleep.changepoints` marked and the detected sleep episodes shaded, so each night's rest timing is visible against the raw counts. Returns a `ggplot` object and never errors.

Usage

```
plot_changepoints(counts, timestamps, ...)
```

Arguments

`counts`	Numeric activity vector (minute epochs recommended).
`timestamps`	POSIXct timestamps, one per value.
`...`	Passed to `sleep.changepoints` (e.g. `thr`).

Value

A `ggplot` object.

References

Chen S, Sun X (2024). "Validating CircaCP: a generic sleep-wake cycle detection algorithm for unlabelled actigraphy data." *Royal Society Open Science*, **11**(5), 231468. doi:10.1098/rsos.231468.

Examples

```
ts <- seq(as.POSIXct("2024-01-01", tz = "UTC"), by = 60, length.out = 5 * 1440)
h  <- as.numeric(format(ts, "%H"))
plot_changepoints(ifelse(h >= 8 & h < 23, 300, 5), ts)
```

plot_chisq *Plot the Chi-Square (Sokolove-Bushell) Periodogram*

Description

Draws the chi-square periodogram from `chi.sq.periodogram`: the Q_P statistic against trial period, with the per-period chi-square significance threshold overlaid as a dashed line. The estimated period (the Q_P peak) is marked and a 24-hour reference line drawn. Unlike the Lomb-Scargle plot, the explicit threshold shows directly whether a rhythm is significant at a given period.

Usage

```
plot_chisq(
  counts,
  timestamps,
  from = 18,
  to = 30,
  alpha = 0.05,
  epoch_length = NULL
)
```

Arguments

counts	Numeric vector of activity counts on a regular epoch grid.
timestamps	A POSIXct vector the same length as counts.
from, to	Period search window in hours (default 18 to 30).
alpha	Significance level for the chi-square threshold (default 0.05).
epoch_length	Epoch length in seconds; inferred from timestamps when NULL.

Value

A ggplot object. Never errors; returns an annotated empty plot on insufficient data.

See Also

plot_periodogram for the Lomb-Scargle spectrum.

---

plot_cosinor_ellipse *Cosinor Amplitude-Acrophase Confidence Ellipse*

Description

Draws the joint confidence ellipse of the cosinor cosine and sine coefficients on the amplitude-acrophase plane, over clock-hour spokes and amplitude rings. The estimate is the vector from the pole, and the rhythm is detectable when the ellipse excludes the pole (Bingham et al. 1982). Returns a ggplot object and never errors.

Usage

```
plot_cosinor_ellipse(counts, timestamps, period = 24, level = 0.95)
```

Arguments

counts	Numeric activity vector.
timestamps	POSIXct timestamps, one per value.
period	Cosinor period in hours (default 24).
level	Confidence level for the ellipse (default 0.95).

Value

A ggplot object.

References

Bingham C, Arbogast B, Cornelissen Guillaume G, Lee JK, Halberg F (1982). "Inferential statistical methods for estimating and comparing cosinor parameters." *Chronobiologia*, **9**(4), 397–439.

Examples

```
set.seed(1)
ts <- seq(as.POSIXct("2024-01-01", tz = "UTC"), by = 60, length.out = 3 * 1440)
h  <- as.numeric(format(ts, "%H"))
counts <- pmax(0, 100 + 80 * cos(2 * pi * (h - 14) / 24) + rnorm(length(h), 0, 25))
plot_cosinor_ellipse(counts, ts)
```

---

plot_cosinor_schematic

Annotated Cosinor Parameter Schematic

Description

A teaching figure: one smooth cosine annotating the MESOR (midline), the amplitude A and double amplitude 2A, the acrophase (clock time of the peak), and the period. Useful as a legend for the cosinor parameters. Returns a ggplot object and never errors.

Usage

```
plot_cosinor_schematic(
  mesor = 100,
  amplitude = 50,
  acrophase = 16,
  period = 24
)
```

Arguments

mesor, amplitude	Midline and amplitude of the illustrated cosine.
acrophase	Clock time of the peak, in hours.
period	Period in hours (default 24).

Value

A ggplot object.

Examples

```
plot_cosinor_schematic()
```

`plot_dfa` *Plot the Detrended Fluctuation Analysis (DFA) Scaling Relationship*

Description

Runs `fractal.dfa` on an activity series and draws the detrended-fluctuation log-log scaling plot: `log10(F(n))` against `log10(n)` (window size). A regression line whose slope is the overall scaling exponent `alpha` is overlaid, and when the analysis splits the scales at a breakpoint the short- (`alpha1`) and long-timescale (`alpha2`) segments are drawn separately. The exponents are annotated on the plot and an interpretive guide appears in the subtitle.

Usage

```
plot_dfa(counts)
```

Arguments

`counts`	Numeric vector of activity counts (minute-level recommended). The longest continuous non-`NA` segment is analyzed internally by `fractal.dfa()`.

Details

DFA quantifies long-range temporal correlations. The scaling exponent is the slope of `log10(F(n))` regressed on `log10(n)`: `alpha` near 0.5 indicates uncorrelated (white) noise, `alpha` near 1.0 indicates 1/f (pink) noise, and `alpha` near 1.5 indicates Brownian (random-walk) noise. The `scales`, `fluctuations`, `alpha`, `alpha1`, `alpha2` and `breakpoint_min` fields returned by `fractal.dfa()` drive the plot directly.

Value

A `ggplot` object: `log10(F(n))` (y) versus `log10(window size)` (x) as points with fitted scaling line(s) and `alpha`/`alpha1`/`alpha2` annotations. On an unusable series (too short, all-`NA`, or constant) an annotated empty `ggplot` is returned. The function never errors.

References

Peng CK, Buldyrev SV, Havlin S, Simons M, Stanley HE, Goldberger AL (1994). "Mosaic organization of DNA nucleotides." *Physical Review E*, **49**(2), 1685–1689. doi:10.1103/PhysRevE.49.1685.

Hu K, Van Someren EJW, Shea SA, Scheer FAJL (2009). "Reduction of scale invariance of activity fluctuations with aging and Alzheimer's disease: Involvement of the circadian pacemaker." *Proceedings of the National Academy of Sciences*, **106**(8), 2490–2494. doi:10.1073/pnas.0806087106.

See Also

fractal.dfa, multiscale.entropy

Examples

```
set.seed(1)
plot_dfa(cumsum(rnorm(5000)))   # Brownian-like, alpha near 1.5
```

---

plot_dichotomy *Dichotomy Index Plot*

Description

Cumulative distributions of the rest-span and active-span activity counts on a log scale, marking the active-span median and the dichotomy index I<O (the share of rest-span counts below that median). Returns a ggplot object and never errors.

Usage

```
plot_dichotomy(counts, rest)
```

Arguments

counts	Numeric activity vector.
rest	Logical (TRUE = rest), or a character state vector as accepted by dichotomy.index. Same length as counts.

Value

A ggplot object.

References

Mormont MC, Waterhouse J, Bleuzen P, Giacchetti S, Jami A, Bogdan A, Lellouch J, Misset JL, Touitou Y, Levi F (2000). "Marked 24-h rest/activity rhythms are associated with better quality of life, better response, and longer survival in patients with metastatic colorectal cancer and good performance status." *Clinical Cancer Research*, **6**(8), 3038–3045.

Examples

```
ts <- seq(as.POSIXct("2024-01-01", tz = "UTC"), by = 60, length.out = 2 * 1440)
h  <- as.numeric(format(ts, "%H"))
plot_dichotomy(ifelse(h >= 23 | h < 7, 5, 300), rest = h >= 23 | h < 7)
```

`plot_emd` *Empirical Mode Decomposition Stack*

Description

Stacks the intrinsic mode functions and residual trend from `circadian.emd`, finest at the top to the trend at the bottom, over a shared time axis, with the circadian IMF highlighted. Returns a `ggplot` object and never errors.

Usage

```
plot_emd(counts, timestamps, ensemble = 1L, epoch_length = 60)
```

Arguments

`counts`	Numeric activity vector.
`timestamps`	POSIXct timestamps, one per value.
`ensemble`	Ensemble size for EEMD (default 1 = plain EMD).
`epoch_length`	Epoch length in seconds (default 60).

Value

A `ggplot` object.

References

Huang NE, Shen Z, Long SR, Wu MC, Shih HH, Zheng Q, Yen NC, Tung CC, Liu HH (1998). "The empirical mode decomposition and the Hilbert spectrum for nonlinear and non-stationary time series analysis." *Proceedings of the Royal Society A*, **454**(1971), 903–995. doi:10.1098/rspa.1998.0193.

Examples

```
ts <- seq(as.POSIXct("2024-01-01", tz = "UTC"), by = 600, length.out = 8 * 144)
th <- as.numeric(difftime(ts, ts[1], units = "hours"))
plot_emd(100 + 60 * cos(2 * pi * th / 24) + 20 * cos(2 * pi * th / 8), ts, epoch_length = 600)
```

`plot_extended_cosinor` *Plot the Extended (Marler) Cosinor Fit on the 24-Hour Activity Profile*

Description

Builds the averaged 24-hour activity profile and overlays two model fits for comparison: the Marler (2006) anti-logistic extended cosinor from `cosinor.antilogistic` (drawn as a bold solid curve) and the ordinary single-component cosinor from `cosinor.analysis` (drawn as a dashed curve). The acrophase (time of peak) is marked with a vertical line and the extended-fit parameters are shown in the subtitle.

Usage

```
plot_extended_cosinor(counts, timestamps, period = 24)
```

Arguments

`counts`	Numeric vector of activity counts (one value per epoch).
`timestamps`	A `POSIXct` vector of timestamps, the same length as `counts`.
`period`	Numeric period of the rhythm in hours passed to `cosinor.antilogistic()` (default `24`).

Details

The profile is the mean activity in each clock-hour bin averaged across all recorded days (bin centres at `hour + 0.5`), matching the profile that `cosinor.analysis()` and `cosinor.antilogistic()` fit internally. The fitted curves are evaluated on a dense 0-24 h grid:

- Extended (Marler): $f(t) = minimum + amplitude \cdot expit(\beta(cos(2\pi(t - acrotime)/T) - \alpha))$, using the `minimum`, `amplitude`, `alpha`, `beta` and `acrotime` returned by `cosinor.antilogistic()`.
- Ordinary: $M + Acos(2\pi(t - acrophase)/T)$, using the `mesor`, `amplitude` and `acrophase` from `cosinor.analysis()`.

The subtitle reports the extended-fit MESOR, amplitude, the `alpha` width-asymmetry and the `beta` steepness when the fit converged.

Value

A `ggplot` object: the averaged hourly profile (points and a light connecting line) over hour-of-day 0-24, with the extended and ordinary cosinor curves overlaid. If the extended fit does not converge the profile and ordinary cosinor are still drawn with a "did not converge" note; on insufficient data an annotated empty `ggplot` is returned. The function never errors.

References

Marler MR, Gehrman P, Martin JL, Ancoli-Israel S (2006). "The sigmoidally transformed cosine curve: a mathematical model for circadian rhythms with symmetric non-sinusoidal shapes." *Statistics in Medicine*, **25**(22), 3893–3904. doi:10.1002/sim.2466.

Cornelissen G (2014). "Cosinor-based rhythmometry." *Theoretical Biology and Medical Modelling*, **11**, 16. doi:10.1186/174246821116.

See Also

`cosinor.antilogistic`, `cosinor.analysis`

Examples

```
ts <- seq(as.POSIXct("2024-01-01 00:00:00"), by = 60, length.out = 1440 * 7)
hour <- as.numeric(format(ts, "%H")) + as.numeric(format(ts, "%M")) / 60
counts <- 50 + 300 * plogis(4 * (cos((hour - 14) * 2 * pi / 24) - 0.2)) +
  rnorm(length(ts), 0, 10)
plot_extended_cosinor(counts, ts)
```

---

plot_lids *LIDS Ultradian Cycle Curve*

Description

Plots the smoothed Locomotor Inactivity During Sleep (LIDS) signal across one sleep period with the best-fit ultradian cosine overlaid, the companion figure to `lids`. The cosine's period and Munich Rhythmicity Index summarise the sleep-cycle oscillation. Returns a `ggplot` object and never errors.

Usage

```
plot_lids(counts, timestamps, sleep_periods, period = 1L, ...)
```

Arguments

`counts`	Numeric activity vector.
`timestamps`	POSIXct timestamps, one per count.
`sleep_periods`	Data frame with `in_bed_time` and `out_bed_time` (as in `lids`).
`period`	Which sleep period to plot (default 1).
`...`	Passed to `lids`.

Value

A `ggplot` object.

References

Winnebeck EC, Fischer D, Leise T, Roenneberg T (2018). "Dynamics and ultradian structure of human sleep in real life." *Current Biology*, **28**(1), 49–59. doi:10.1016/j.cub.2017.11.063.

Examples

```
ts <- seq(as.POSIXct("2024-01-01 22:00", tz = "UTC"), by = 60, length.out = 480)
sp <- data.frame(in_bed_time = ts[1], out_bed_time = ts[480])
plot_lids(50 + 45 * cos(2 * pi * seq_along(ts) / 110), ts, sp)
```

---

plot_mfdfa *Multifractal DFA Spectrum*

Description

Two panels from `mfdfa`: the generalized Hurst exponent h(q) against q (flat = monofractal, decreasing = multifractal, with h(2) marked) and the singularity spectrum f(alpha) (the arch whose width measures multifractality). Returns a `ggplot` object and never errors.

Usage

```
plot_mfdfa(counts, q_values = seq(-5, 5, by = 0.5), detrend_order = 1L)
```

Arguments

`counts`	Numeric activity vector.
`q_values`	Moment orders to evaluate (default `seq(-5, 5, by = 0.5)`).
`detrend_order`	Within-window polynomial detrend order (default 1).

Value

A `ggplot` object.

References

Kantelhardt JW, Zschiegner SA, Koscielny-Bunde E, Havlin S, Bunde A, Stanley HE (2002). "Multifractal detrended fluctuation analysis of nonstationary time series." *Physica A: Statistical Mechanics and its Applications*, **316**(1-4), 87–114. doi:10.1016/S0378437(02)013833.

Examples

```
set.seed(1)
plot_mfdfa(cumsum(rnorm(8000)))
```

`plot_mse` *Multiscale Entropy Curve*

Description

Sample entropy from `multiscale.entropy` against the coarse-graining scale, with the complexity index (area) shaded and the across-scale slope drawn. A curve that holds its entropy across scales marks a complex signal; one that falls is closer to noise. Returns a `ggplot` object and never errors.

Usage

```
plot_mse(counts, scales = 1:20, m = 2L, r = 0.15)
```

Arguments

`counts`	Numeric activity vector.
`scales`	Integer coarse-graining scales (default `1:20`).
`m`	Embedding dimension for sample entropy (default 2).
`r`	Tolerance as a fraction of the series SD (default 0.15).

Value

A `ggplot` object.

References

Costa M, Goldberger AL, Peng C (2002). "Multiscale entropy analysis of complex physiologic time series." *Physical Review Letters*, **89**(6), 068102. doi:10.1103/PhysRevLett.89.068102.

Examples

```
set.seed(1)
plot_mse(rnorm(3000))
```

`plot_multicomponent` *Single Cosine vs Multicomponent Cosinor Fit*

Description

Overlays the averaged daily profile with the single 24-hour cosine and the selected multi-harmonic cosinor (Cornelissen 2014), showing the structure a single symmetric cosine cannot follow. The number of harmonics and the fit are taken from `cosinor.multicomponent`; the curves are refit on the hour-of-day profile so they align with the plotted points. Returns a `ggplot` object and never errors.

Usage

```
plot_multicomponent(counts, timestamps, period = 24, max_harmonics = 3)
```

Arguments

`counts`	Numeric activity vector.
`timestamps`	POSIXct timestamps, one per value.
`period`	Fundamental period in hours (default 24).
`max_harmonics`	Largest number of harmonics to consider (default 3).

Value

A `ggplot` object.

References

Cornelissen G (2014). "Cosinor-based rhythmometry." *Theoretical Biology and Medical Modelling*, **11**, 16. doi:10.1186/174246821116.

Examples

```
set.seed(2)
ts <- seq(as.POSIXct("2024-01-01", tz = "UTC"), by = 60, length.out = 4 * 1440)
h  <- as.numeric(format(ts, "%H"))
siesta <- pmax(0, 120 + 70 * cos(2 * pi * (h - 14) / 24) +
  55 * cos(2 * pi * 2 * (h - 14) / 24) + rnorm(length(ts), 0, 12))
plot_multicomponent(siesta, ts)
```

plot_multiscale *Multiscale IS and IV Profiles*

Description

Interdaily stability and intradaily variability recomputed across epoch lengths, with the averaged ISm and IVm marked. Returns a ggplot object and never errors.

Usage

```
plot_multiscale(counts, timestamps)
```

Arguments

counts	Numeric activity vector.
timestamps	POSIXct timestamps, one per value.

Value

A ggplot object.

References

Goncalves BSB, Cavalcanti PRA, Tavares GR, Campos TF, Araujo JF (2014). "Nonparametric methods in actigraphy: an update." *Sleep Science*, **7**(3), 158–164. doi:10.1016/j.slsci.2014.09.013.

Examples

```
ts <- seq(as.POSIXct("2024-01-01", tz = "UTC"), by = 60, length.out = 4 * 1440)
h  <- as.numeric(format(ts, "%H"))
plot_multiscale(pmax(0, 100 + 80 * cos(2 * pi * (h - 14) / 24)), ts)
```

plot_periodogram *Plot the Lomb-Scargle Periodogram with the Endogenous Period*

Description

Computes the full Lomb-Scargle periodogram of an activity series over a period search window and plots spectral power against period (in hours). The dominant endogenous period (tau) estimated by circadian.period is marked with a labelled vertical line, and a dashed reference line is drawn at 24 hours so the deviation of the biological clock from exactly one solar day is visible.

Usage

```
plot_periodogram(
  counts,
  timestamps,
  from = 18,
  to = 30,
  ofac = 4,
  alpha = 0.05
)
```

Arguments

counts	Numeric vector of activity counts (minute-level recommended). NA values (e.g. non-wear epochs) are dropped together with their timestamps before estimation.
timestamps	A POSIXct vector (or anything coercible by as.numeric) of epoch timestamps, the same length as counts.
from	Numeric. Lower bound of the period search window, in hours (default 18).
to	Numeric. Upper bound of the period search window, in hours (default 30).
ofac	Integer oversampling factor for the period grid. Higher values give a finer grid (default 4).
alpha	Significance level for the Baluev false-alarm threshold (default 0.05).

Details

The full standard-normalized Lomb-Scargle spectrum over the period window is computed by the package's own estimator (the same one behind circadian.period); the peak period tau, its Baluev p_value, and the false-alarm threshold line all come from that function, so the highlighted peak and threshold match the reported values. The Lomb-Scargle periodogram is the least-squares spectral estimator for unevenly sampled series and is therefore appropriate for gappy actigraphy data, which an FFT cannot accommodate.

Value

A ggplot object: Lomb-Scargle power (y) versus period in hours (x), with the peak period and the 24 h reference annotated. On insufficient data (all-NA, constant, or fewer than about 2 days of span) a ggplot carrying a centred "Insufficient data for periodogram" annotation is returned instead; the function never errors.

References

Lomb NR (1976). "Least-squares frequency analysis of unequally spaced data." *Astrophysics and Space Science*, **39**(2), 447–462. doi:10.1007/BF00648343.

Scargle JD (1982). "Studies in astronomical time series analysis. II. Statistical aspects of spectral analysis of unevenly spaced data." *The Astrophysical Journal*, **263**, 835–853. doi:10.1086/160554.

Ruf T (1999). "The Lomb-Scargle periodogram in biological rhythm research: analysis of incomplete and unequally spaced time-series." *Biological Rhythm Research*, **30**(2), 178–201. doi:10.1076/brhm.30.2.178.1422.

See Also

circadian.period, plot_extended_cosinor

Examples

```
t_hours <- seq(0, 7 * 24 - 1 / 60, by = 1 / 60)
ts <- as.POSIXct("2024-01-01 00:00:00") + t_hours * 3600
counts <- 100 + 80 * cos(2 * pi * (t_hours - 8) / 24) +
  rnorm(length(t_hours), 0, 5)
plot_periodogram(counts, ts)
```

---

plot_period_ci *Bootstrap Period Confidence Interval*

Description

Shows the distribution of bootstrap replicate periods from period.ci with the point estimate and the confidence-interval band marked, so the period estimate is read together with its uncertainty. Returns a ggplot object and never errors.

Usage

```
plot_period_ci(
  counts,
  timestamps,
  from = 18,
  to = 30,
  level = 0.95,
  n_boot = 200,
  seed = NULL
)
```

Arguments

counts	Numeric activity vector.
timestamps	POSIXct timestamps, one per value.
from, to	Period search window in hours (default 18, 30).
level	Confidence level (default 0.95).
n_boot	Bootstrap replicates (default 200).
seed	Optional RNG seed for reproducibility.

Value

A ggplot object.

References

Kunsch HR (1989). "The jackknife and the bootstrap for general stationary observations." *The Annals of Statistics*, **17**(3), 1217–1241. doi:10.1214/aos/1176347265.

Politis DN, Romano JP (1992). "A circular block-resampling procedure for stationary data." In LePage R, Billard L (eds.), *Exploring the Limits of Bootstrap*, 263–270. Wiley, New York.

Examples

```
ts <- seq(as.POSIXct("2024-01-01", tz = "UTC"), by = 60, length.out = 7 * 1440)
h  <- as.numeric(format(ts, "%H"))
plot_period_ci(100 + 60 * cos(2 * pi * (h - 8) / 24) + rnorm(length(ts), 0, 20), ts)
```

`plot_phase_rose` *Circular Phase Rose*

Description

Plots daily phase markers (acrophases, onsets, L5/M10 times) as a rose diagram around the 24-hour clock, with the mean resultant vector drawn from the centre and the Rayleigh and Hermans-Rasson results annotated. This is the companion figure to `phase.concentration`. Sector area, not radius, encodes the count (the radius uses a square-root scale) so a wide sector is not read as a large one. Returns a `ggplot` object and never errors.

Usage

```
plot_phase_rose(times_h, period = 24, binwidth = 1)
```

Arguments

`times_h`	Numeric vector of clock times (hours), one per day.
`period`	Period the times wrap on, in hours (default 24).
`binwidth`	Sector width in hours (default 1).

Value

A `ggplot` object.

References

Fisher NI (1993). *Statistical Analysis of Circular Data*. Cambridge University Press. doi:10.1017/CBO9780511564345.

Landler L, Ruxton GD, Malkemper EP (2019). "The Hermans-Rasson test as a powerful alternative to the Rayleigh test for circular statistics in biology." *BMC Ecology*, **19**, 30. doi:10.1186/s12898-01902468.

Examples

```
set.seed(1)
plot_phase_rose(23 + stats::rnorm(30, 0, 1))
```

`plot_profile` *Average-Day Profile with the L5 and M10 Windows*

Description

The averaged daily activity profile (mean across days with a one-standard- deviation band), with the least-active 5-hour window L5 and the most-active 10-hour window M10 marked on a window track beneath the curve. Returns a `ggplot` object and never errors.

Usage

```
plot_profile(counts, timestamps, bin_min = 30)
```

Arguments

`counts`	Numeric activity vector.
`timestamps`	POSIXct timestamps, one per value.
`bin_min`	Profile resolution in minutes (default 30).

Value

A `ggplot` object.

References

Van Someren EJW, Swaab DF, Colenda CC, Cohen W, McCall WV, Rosenquist PB (1999). "Bright light therapy: improved sensitivity to its effects on rest-activity rhythms in Alzheimer patients by application of nonparametric methods." *Chronobiology International*, **16**(4), 505–518. doi:10.3109/07420529908998724.

Examples

```
ts <- seq(as.POSIXct("2024-01-01", tz = "UTC"), by = 60, length.out = 7 * 1440)
h  <- as.numeric(format(ts, "%H"))
plot_profile(pmax(0, 100 + 80 * cos(2 * pi * (h - 14) / 24)), ts)
```

`plot_raw_metrics` *Raw-Acceleration Epoch Metrics Profile*

Description

Plots the per-epoch ENMO, MAD, and z-angle from `raw.metrics` as a faceted time series, a quality-control view of the gravity-preserving signals. Returns a `ggplot` object and never errors.

Usage

```
plot_raw_metrics(x, epoch_length = 60, ...)
```

Arguments

`x`	A path to a raw file or a raw data frame (as in `raw.metrics`).
`epoch_length`	Epoch length in seconds (default 60).
`...`	Passed to `raw.metrics` (e.g. `metrics`, `calibrate`).

Value

A `ggplot` object.

References

van Hees VT, Gorzelniak L, Dean Leon EC, Eder M, Pias M, Taherian S, Ekelund U, Renstrom F, Franks PW, Horsch A, Brage S (2013). "Separating movement and gravity components in an acceleration signal and implications for the assessment of human daily physical activity." *PLoS ONE*, **8**(4), e61691. doi:10.1371/journal.pone.0061691.

Vaha-Ypya H, Vasankari T, Husu P, Suni J, Sievanen H (2015). "A universal, accurate intensity-based classification of different physical activities using raw data of accelerometer." *Clinical Physiology and Functional Imaging*, **35**(1), 64–70. doi:10.1111/cpf.12127.

van Hees VT, Sabia S, Anderson KN, Denton SJ, Oliver J, Catt M, Abell JG, Kivimaki M, Trenell MI, Singh-Manoux A (2015). "A novel, open access method to assess sleep duration using a wrist-worn accelerometer." *PLoS ONE*, **10**(11), e0142533. doi:10.1371/journal.pone.0142533.

Examples

```
plot_raw_metrics(example_raw(days = 1))
```

plot_rest_comparison *Rest-Detector Comparison Strip*

Description

Runs the four rest/sleep detectors on one recording and stacks their detected rest bands over the activity series on a shared time axis, so the user can see that the differing bout counts reflect different questions (main night vs every bout vs latent state), not contradictory accuracy. Returns a `ggplot` object and never errors.

Usage

```
plot_rest_comparison(counts, timestamps)
```

Arguments

`counts`	Numeric activity vector.
`timestamps`	POSIXct timestamps, one per value.

Value

A `ggplot` object.

See Also

`sleep.changepoints`, `rest.periods`, `rest.crespo`, `rest.hmm`

Examples

```
ts <- seq(as.POSIXct("2024-01-01", tz = "UTC"), by = 60, length.out = 4 * 1440)
h  <- as.numeric(format(ts, "%H"))
plot_rest_comparison(ifelse(h >= 23 | h < 7, 5, 300), ts)
```

plot_spt *z-Angle Sleep-Period-Time Detection*

Description

Plots the z-angle of a raw recording with the HDCZA sleep-period-time window(s) shaded and non-wear marked, the companion figure to `rest.spt`. Computes the angle, non-wear, and SPT window from the raw input. Returns a `ggplot` object and never errors.

Usage

```
plot_spt(x, epoch_length = 5)
```

Arguments

x	A path to a raw file or a raw data frame (as in raw.metrics).
epoch_length	Epoch length in seconds (default 5, matching rest.spt).

Value

A ggplot object.

References

van Hees VT, Sabia S, Jones SE, Wood AR, Anderson KN, Kivimaki M, Frayling TM, Pack AI, Bucan M, Trenell MI, Mazzotti DR, Gehrman PR, Singh-Manoux BA, Weedon MN (2018). "Estimating sleep parameters using an accelerometer without sleep diary." *Scientific Reports*, **8**, 12975. doi:10.1038/s4159801831266z.

Examples

```
plot_spt(example_raw(days = 2))
```

---

plot_ssa_wcor *SSA w-Correlation Matrix*

Description

Plots the weighted-correlation matrix of the leading SSA elementary components from circadian.ssa. Bright off-diagonal 2x2 blocks are oscillatory pairs (the grouping aid of Golyandina 2013); a single bright cell is the trend and a diffuse block is noise. Returns a ggplot object and never errors.

Usage

```
plot_ssa_wcor(counts, timestamps, window_hours = 48)
```

Arguments

counts	Numeric activity vector.
timestamps	POSIXct timestamps, one per value.
window_hours	SSA window length in hours (default 48).

Value

A ggplot object.

References

Golyandina N, Zhigljavsky A (2013). *Singular Spectrum Analysis for Time Series*, SpringerBriefs in Statistics. Springer. doi:10.1007/9783642349133.

Examples

```
ts <- seq(as.POSIXct("2024-01-01", tz = "UTC"), by = 600, length.out = 8 * 144)
th <- as.numeric(difftime(ts, ts[1], units = "hours"))
plot_ssa_wcor(100 + 60 * cos(2 * pi * th / 24), ts)
```

---

plot_transitions *Rest-Activity Transition Curves*

Description

The rest-to-active and active-to-rest transition hazards against bout length, with a LOWESS fit and the sustained rates kRA and kAR marked. Returns a ggplot object and never errors.

Usage

```
plot_transitions(counts, threshold = 1, frac = 0.3)
```

Arguments

counts	Numeric activity vector.
threshold	Counts at or above which an epoch is active (default 1).
frac	LOWESS span (default 0.3).

Value

A ggplot object.

References

Lim ASP, Yu L, Costa MD, Buchman AS, Bennett DA, Leurgans SE, Saper CB (2011). "Quantification of the fragmentation of rest-activity patterns in elderly individuals using a state transition analysis." *Sleep*, **34**(11), 1569–1581. doi:10.5665/sleep.1400.

Examples

```
set.seed(1)
plot_transitions(as.integer(stats::runif(8000) < 0.1) * 100)
```

`plot_wavelet` *Wavelet Power Scalogram*

Description

Draws the Morlet wavelet power surface from `circadian.wavelet`: time on the x-axis, period (hours, log scale) on the y, scale-rectified power as the fill, with the cone of influence faded out, the per-time dominant-period ridge traced, and the 24-hour reference marked. Returns a `ggplot` object and never errors.

Usage

```
plot_wavelet(counts, timestamps, epoch_length = 60)
```

Arguments

`counts`	Numeric activity vector.
`timestamps`	POSIXct timestamps, one per value.
`epoch_length`	Epoch length in seconds (default 60).

Value

A `ggplot` object.

References

Torrence C, Compo GP (1998). "A practical guide to wavelet analysis." *Bulletin of the American Meteorological Society*, **79**(1), 61–78. doi:10.1175/15200477(1998)079<0061:APGTWA>2.0.CO;2.

Leise TL (2013). "Wavelet analysis of circadian and ultradian behavioral rhythms." *Journal of Circadian Rhythms*, **11**, 5. doi:10.1186/17403391115.

Examples

```
ts <- seq(as.POSIXct("2024-01-01", tz = "UTC"), by = 600, length.out = 8 * 144)
th <- as.numeric(difftime(ts, ts[1], units = "hours"))
plot_wavelet(100 + 80 * cos(2 * pi * th / 24), ts, epoch_length = 600)
```

`population.cosinor` *Population-Mean Cosinor (Bingham)*

Description

Pools single-subject cosinor fits into a group-mean rhythm with confidence intervals, following Bingham et al. (1982). Fits each subject's averaged 24-hour profile with the same weighted-least-squares engine as `cosinor.analysis`, averages the linearized cos/sin coefficients across subjects, and returns the group MESOR, amplitude, and acrophase with Bingham confidence intervals.

Usage

```
population.cosinor(
  activity,
  timestamps,
  subject,
  group = NULL,
  period = 24,
  level = 0.95,
  min_valid_hours = 12
)
```

Arguments

`activity`	Numeric activity vector with all subjects stacked together.
`timestamps`	POSIXct timestamps, one per value.
`subject`	Subject identifier, one per value.
`group`	Optional group identifier, one per value; when supplied a population cosinor is returned for each group.
`period`	Rhythm period in hours (default 24).
`level`	Confidence level (default 0.95).
`min_valid_hours`	Minimum profile hours for a subject to be included (default 12).

Value

An object of class `actiRhythm_population_cosinor` (group MESOR, amplitude, acrophase with Bingham CIs and a `conf_interval_valid` flag), or a named list of them (class `actiRhythm_population_cosinor_list`) when `group` is supplied.

References

Bingham C, Arbogast B, Cornelissen Guillaume G, Lee JK, Halberg F (1982). "Inferential statistical methods for estimating and comparing cosinor parameters." *Chronobiologia*, **9**(4), 397–439.

Examples

```
set.seed(1)
hrs <- 0:23
act <- ts <- subj <- NULL
for (i in 1:6) {
  y <- 100 + 40 * cos(2 * pi * (hrs - (8 + i / 3)) / 24) + rnorm(24, 0, 4)
  act <- c(act, y); subj <- c(subj, rep(paste0("S", i), 24))
  ts <- c(ts, as.POSIXct("2024-01-01", tz = "UTC") + hrs * 3600)
}
population.cosinor(act, as.POSIXct(ts, tz = "UTC", origin = "1970-01-01"), subj)
```

---

print.actiRhythm_circadian

Print Method for Circadian Rhythm Results

Description

Print Method for Circadian Rhythm Results

Usage

```
## S3 method for class 'actiRhythm_circadian'
print(x, ...)
```

Arguments

x	Object of class 'actiRhythm_circadian'
...	Additional arguments (unused)

Value

Invisibly returns x.

---

print.actiRhythm_cosinor

Print method for cosinor analysis

Description

Print method for cosinor analysis

Usage

```
## S3 method for class 'actiRhythm_cosinor'
print(x, ...)
```

Arguments

x	actiRhythm_cosinor object
...	Additional arguments (ignored)

Value

Invisibly returns x.

---

print.actiRhythm_cosinor_extended *Print method for extended cosinor analysis*

Description

Print method for extended cosinor analysis

Usage

```
## S3 method for class 'actiRhythm_cosinor_extended'
print(x, ...)
```

Arguments

x	actiRhythm_cosinor_extended object
...	Additional arguments (ignored)

Value

Invisibly returns x.

---

print.actiRhythm_dfa *Print Method for DFA Results*

Description

Print Method for DFA Results

Usage

```
## S3 method for class 'actiRhythm_dfa'
print(x, ...)
```

Arguments

x	Object of class "actiRhythm_dfa".
...	Additional arguments (unused).

Value

Invisibly returns `x`.

---

print.actiRhythm_mse *Print Method for Multiscale Entropy Results*

Description

Print Method for Multiscale Entropy Results

Usage

```
## S3 method for class 'actiRhythm_mse'
print(x, ...)
```

Arguments

`x`	Object of class `"actiRhythm_mse"`.
`...`	Additional arguments (unused).

Value

Invisibly returns `x`.

---

raw.metrics *Raw-Acceleration Epoch Metrics (ENMO, MAD, z-Angle)*

Description

Reads a raw accelerometer file and returns per-epoch raw activity and posture metrics, the gravity-preserving signals that counts cannot represent: ENMO (Euclidean Norm Minus One, a raw activity metric), MAD (Mean Amplitude Deviation), and the z-angle (arm/posture angle). Auto-calibration (van Hees 2014) is applied first by default. Requires the relevant raw reader (**read.gt3x** for `.gt3x`, **GGIRread** for `.cwa`/`.bin`).

Usage

```
raw.metrics(
  x,
  device = "auto",
  epoch = 60,
  metrics = c("ENMO", "MAD", "anglez"),
  calibrate = TRUE,
  tz = "UTC"
)
```

Arguments

x	A path to a raw file (.gt3x, .cwa, .bin) or a raw data frame with x/y/z columns in g and an fs attribute (e.g. from example_raw or your own device).
device	One of "auto", "gt3x", "axivity", "geneactiv" (default "auto", inferred from the extension; used only when x is a file path).
epoch	Epoch length in seconds (default 60).
metrics	Which metrics to return; any of "ENMO", "MAD", "anglez" (default all three).
calibrate	Apply van Hees auto-calibration first (default TRUE).
tz	Time zone for the timestamps (default "UTC").

Value

A data frame with time and the requested metrics (ENMO and MAD in mg, anglez in degrees), one row per epoch. The calibration result is attached as the "calibration" attribute.

References

van Hees VT, Gorzelniak L, Dean Leon EC, Eder M, Pias M, Taherian S, Ekelund U, Renstrom F, Franks PW, Horsch A, Brage S (2013). "Separating movement and gravity components in an acceleration signal and implications for the assessment of human daily physical activity." *PLoS ONE*, **8**(4), e61691. doi:10.1371/journal.pone.0061691.

Vaha-Ypya H, Vasankari T, Husu P, Suni J, Sievanen H (2015). "A universal, accurate intensity-based classification of different physical activities using raw data of accelerometer." *Clinical Physiology and Functional Imaging*, **35**(1), 64–70. doi:10.1111/cpf.12127.

van Hees VT, Sabia S, Anderson KN, Denton SJ, Oliver J, Catt M, Abell JG, Kivimaki M, Trenell MI, Singh-Manoux A (2015). "A novel, open access method to assess sleep duration using a wrist-worn accelerometer." *PLoS ONE*, **10**(11), e0142533. doi:10.1371/journal.pone.0142533.

See Also

auto.calibrate, circadian.raw, rest.spt, example_raw

Examples

```
# On a synthetic raw recording (no file needed); pass a path for a real file

m <- raw.metrics(example_raw(days = 1), epoch = 60)
head(m)
```

`read.actigraph.csv` *Read an ActiGraph (ActiLife) Epoch CSV*

Description

Reads an ActiLife epoch (count) CSV export. The metadata header is parsed for the start time and epoch length, the count columns are recognized by name or position, and a regular timestamp grid is built. The result is a data frame with a POSIXct `timestamp` and the available count columns, the same shape as `agd.counts`, so it feeds the analysis functions directly.

Usage

```
read.actigraph.csv(
  filepath,
  tz = "UTC",
  date_format = "%m/%d/%Y",
  epoch_length = 60
)
```

Arguments

`filepath`	Path to the ActiLife CSV.
`tz`	Time zone for the timestamps (default `"UTC"`; ActiLife stores local clock time without a zone).
`date_format`	Date format of the header Start Date (default `"%m/%d/%Y"`, the ActiLife M/d/yyyy default).
`epoch_length`	Fallback epoch length in seconds if the header lacks an Epoch Period line (default 60).

Value

A data frame with `timestamp` (POSIXct) and the count columns present (`axis1`, `axis2`, `axis3`, `vm`, `steps`, `lux`).

See Also

`counts.from.data.frame`, `read.agd`

`read.agd` *Read ActiGraph .agd File*

Description

Reads epoch-level AGD files pre-processed in ActiLife. AGD files hold activity counts already computed from raw acceleration data.

Usage

```
read.agd(
  filepath,
  include_sleep = TRUE,
  include_wear_time = TRUE,
  verbose = TRUE
)
```

Arguments

`filepath`	Path to .agd file
`include_sleep`	Logical. Include sleep analysis data if available? (default: TRUE)
`include_wear_time`	Logical. Include wear time validation data if available? (default: TRUE)
`verbose`	Logical. Print progress messages? (default: TRUE)

Details

AGD files are SQLite databases created by ActiLife software. They contain pre-processed activity counts at user-specified epoch lengths (e.g., 60 seconds).

Value

List containing:

- `data` - Data frame with epoch-level activity counts (axis1, axis2, axis3, steps, etc.)
- `settings` - Data frame with device and subject settings
- `sleep` - Sleep periods detected by ActiLife (if available and requested)
- `awakenings` - Awakening events during sleep (if available and requested)
- `wear_time` - Wear time validation bouts (if available and requested)
- `capsense` - Capacitive sensor data for on-body detection (if available)
- `tables` - Character vector of all tables in the AGD file

read.raw *Activity Counts from a Raw Accelerometer File (Any Supported Brand)*

Description

A single entry point that dispatches to the brand-specific reader by file extension (or an explicit `device`) and returns ActiGraph-equivalent counts: ActiGraph `.gt3x`, Axivity `.cwa`, or GENEActiv `.bin`.

Usage

```
read.raw(
  path,
  device = c("auto", "gt3x", "axivity", "geneactiv"),
  epoch = 60,
  lfe = FALSE,
  tz = "UTC"
)
```

Arguments

`path`	Path to a raw file.
`device`	One of `"auto"` (infer from the extension), `"gt3x"`, `"axivity"`, `"geneactiv"`.
`epoch`	Epoch length in seconds (default 60).
`lfe`	Use the low-frequency extension filter (default `FALSE`).
`tz`	Time zone for the timestamps (default `"UTC"`).

Value

Data frame with `time`, `axis1`, `axis2`, `axis3` and `vm`, one row per epoch. Counts from non-ActiGraph devices are an approximation; see `axivity.counts` / `geneactiv.counts`.

See Also

`gt3x.counts`, `axivity.counts`, `geneactiv.counts`

`residual.spectrum` *Residual Circadian Spectrum*

Description

Removes the fitted cosinor mean and estimates the spectrum of what is left, the residual circadian spectrum, integrating it into frequency bands. It measures the ultradian and noise structure that the cosinor does not fit (Krafty et al. 2019), so two recordings with the same 24-hour rhythm but different residual fragmentation are told apart.

Usage

```
residual.spectrum(
  counts,
  timestamps,
  period = 24,
  bands = list(ultradian = c(2, 8), high_freq = c(0.5, 2))
)
```

Arguments

`counts`	Numeric activity vector.
`timestamps`	POSIXct timestamps, one per value.
`period`	Cosinor period in hours (default 24).
`bands`	Named list of `c(low_hours, high_hours)` period bands for the residual power; defaults to ultradian (2-8 h) and high-frequency (0.5-2 h).

Value

An object of class `actiRhythm_rcs`: the residual variance, a per-band power table, and the spectrum. Never errors.

References

Krafty RT, Fu H, Graves JL, Bruce SA, Hall MH, Smagula SF (2019). "Measuring variability in rest-activity rhythms from actigraphy with application to characterizing symptoms of depression." *Statistics in Biosciences*, **11**, 314–333. doi:10.1007/s12561018092302.

Examples

```
ts <- seq(as.POSIXct("2024-01-01", tz = "UTC"), by = 600, length.out = 6 * 144)
th <- as.numeric(difftime(ts, ts[1], units = "hours"))
residual.spectrum(100 + 80 * cos(2 * pi * th / 24) + 20 * sin(2 * pi * th / 4), ts,
                  period = 24)
```

`rest.activity.fragmentation` *Rest-Activity Bout Fragmentation*

Description

Summarizes how broken up the rest-activity rhythm is, from a per-epoch rest/active state: the mean and median rest and active bout durations, the number of state transitions, and transitions per day. These add a bout-length view of fragmentation to the transition probabilities of `state.transitions` (kRA/kAR) (Lim et al. 2011). It does not cover sedentary-bout distribution metrics (Gini, power law, hazard).

Usage

```
rest.activity.fragmentation(state, timestamps, epoch_length = 60)
```

Arguments

`state`	Per-epoch state: a logical vector (TRUE = active) or a character vector where `"R"`/`"S"`/`"sleep"`/`"rest"` mark rest.
`timestamps`	POSIXct timestamps, one per value.
`epoch_length`	Epoch length in seconds (default 60).

Value

An object of class `actiRhythm_rafrag`: mean/median rest and active bout durations (minutes), bout counts, transition count, and transitions per day. Never errors.

References

Lim ASP, Yu L, Costa MD, Buchman AS, Bennett DA, Leurgans SE, Saper CB (2011). "Quantification of the fragmentation of rest-activity patterns in elderly individuals using a state transition analysis." *Sleep*, **34**(11), 1569–1581. doi:10.5665/sleep.1400.

See Also

`state.transitions`

Examples

```
ts <- seq(as.POSIXct("2024-01-01", tz = "UTC"), by = 60, length.out = 3 * 1440)
h  <- as.numeric(format(ts, "%H"))
rest.activity.fragmentation(h >= 7 & h < 23, ts)
```

`rest.crespo` *Crespo Rest/Activity Period Detection*

Description

Detects the main rest and activity periods across a recording with the Crespo algorithm (Crespo et al. 2012): a rank-order (median) filter at the `alpha` (~rest-length) scale, an `alpha`/24h percentile threshold, and a separate minutes-scale morphology pass that turn activity counts into a binary rest/activity series read bout by bout from its transitions. The wide smoothing window suppresses short within-period transitions, so it reports the main daily rest rather than every nap.

Usage

```
rest.crespo(
  counts,
  timestamps,
  epoch_length = 60,
  zeta = 15,
  zeta_r = 30,
  zeta_a = 2,
  t = 0.33,
  alpha = 8 * 3600,
  beta = 1 * 3600
)
```

Arguments

`counts`	Numeric activity vector (minute epochs recommended).
`timestamps`	POSIXct timestamps, one per value.
`epoch_length`	Epoch length in seconds (default 60).
`zeta`, `zeta_r`, `zeta_a`	Maximum valid consecutive-zero run, in epochs, for the global pre-conditioning, within rest segments, and within active segments (defaults 15, 30, 2). Longer zero runs are treated as non-wear.
`t`	Quantile of the counts used to replace non-wear epochs (default 0.33).
`alpha`	Expected daily rest length, in seconds (default 28800, 8 hours); sets the rank-order smoothing window (Eq 4), the threshold percentile (alpha / 24 h), and the minimum data required.
`beta`	Morphology scale, in seconds (default 3600, 1 hour); sets the structuring-element size that consolidates the thresholded series.

Value

An object of class `actiRhythm_crespo`: a `rest_periods` data frame (one row per bout, with onset, offset, and duration), the per-epoch `rest_state` ("R"/"A"), and per-bout counts. The function never errors; with no resolvable bout it returns an empty `rest_periods`.

References

Crespo C, Aboy M, Fernandez JR, Mojon A (2012). "Automatic identification of activity-rest periods based on actigraphy." *Medical & Biological Engineering & Computing*, **50**(4), 329–340. doi:10.1007/s115170120875y.

Hammad G, Reyt M, Beliy N, Baillet M, Deantoni M, Lesoinne A, Muto V, Schmidt C (2021). "pyActigraphy: open-source python package for actigraphy data visualization and analysis." *PLOS Computational Biology*, **17**(10), e1009514. doi:10.1371/journal.pcbi.1009514.

See Also

rest.periods, sleep.changepoints

Examples

```
# Two nights of rest with a daytime nap between them
ts <- seq(as.POSIXct("2024-01-01", tz = "UTC"), by = 60, length.out = 3 * 1440)
h  <- as.numeric(format(ts, "%H")) + as.numeric(format(ts, "%M")) / 60
day <- as.integer(format(ts, "%j"))
counts <- ifelse(h >= 23 | h < 7, 5, 300)
counts[h >= 14 & h < 15 & day == min(day) + 1L] <- 5
rest.crespo(counts, ts)
```

---

rest.hmm *State-Space (Hidden Markov) Rest-Activity Model*

Description

Fits an unsupervised Gaussian hidden Markov model to the activity counts, inferring latent rest and active states and the rhythm with which the subject moves between them (Huang et al. 2018). A threshold scorer labels each epoch independently; the HMM uses the persistence of states. Its decoded path gives a 24-hour state-occupancy profile, the probability of being at rest across the day.

Usage

```
rest.hmm(
  counts,
  timestamps,
  states = 2L,
  transform = c("sqrt", "log", "none"),
  decode = c("posterior", "viterbi"),
  max_iter = 200L,
  tol = 1e-06,
  n_starts = 5L,
  seed = NULL
)
```

Arguments

counts	Numeric activity vector (a coarse epoch is recommended for speed).
timestamps	POSIXct timestamps, one per value.
states	Number of hidden states (default 2 = rest/active; 3 adds a moderate state). Huang et al. (2018) fix three states; the BIC reported here can guide the choice.
transform	Variance-stabilizing transform of the counts: "sqrt" (default), "log", or "none". Huang et al. fit on 5-minute averaged counts, so a coarse epoch is recommended.
decode	State decoding: "posterior" (default, the local decoding of Huang et al. 2018) or "viterbi" (the global most-likely path).
max_iter, tol	EM iteration cap and log-likelihood tolerance.
n_starts	Random restarts; the highest-likelihood fit is kept (default 5).
seed	Optional seed for the restarts.

Value

An object of class actiRhythm_hmm: the per-state emission means and SDs, the transition matrix, the decoded state_path and a sleep_state ("S"/"W") vector, a 24-hour profile of the mean posterior rest probability, and the log-likelihood with AIC/BIC. Never errors.

References

Huang Q, Cohen D, Komarzynski S, Li X, Innominato P, Levi F, Finkenstadt B (2018). "Hidden Markov models for monitoring circadian rhythmicity in telemetric activity data." *Journal of the Royal Society Interface*, **15**(139), 20170885. doi:10.1098/rsif.2017.0885.

See Also

sleep.changepoints, sleep.cole.kripke

Examples

```
ts <- seq(as.POSIXct("2024-01-01", tz = "UTC"), by = 600, length.out = 6 * 144)
h  <- as.numeric(format(ts, "%H"))
counts <- ifelse(h >= 23 | h < 7, 2, 250) + pmax(0, stats::rnorm(length(ts), 0, 10))
rest.hmm(counts, ts)
```

---

rest.periods *Consolidated Rest-Period Detection (Roenneberg / MASDA)*

Description

Detects all consolidated rest bouts across a recording with the Roenneberg consolidation algorithm (the Munich Actimetry Sleep Detection Algorithm). Each epoch is compared to a fraction of its own 24-hour activity trend, runs below that threshold seed candidate rest bouts, and a correlation procedure grows each seed into a consolidated bout. Unlike sleep.changepoints (one nightly bout per cycle), this returns any number of bouts, including daytime naps and fragmented or polyphasic rest.

Usage

```
rest.periods(
  counts,
  timestamps,
  epoch_length = 60,
  trend_period = 86400,
  min_trend_period = 43200,
  threshold = 0.15,
  min_seed_period = 1800,
  max_test_period = 43200,
  r_consec_below = 1800,
  nap_max_minutes = 180
)
```

Arguments

`counts`	Numeric activity vector (minute epochs recommended).
`timestamps`	POSIXct timestamps, one per value.
`epoch_length`	Epoch length in seconds (default 60).
`trend_period`	Moving-average window for the activity trend, in seconds (default 86400, 24 hours).
`min_trend_period`	Minimum non-missing seconds required in the trend window (default 43200, 12 hours).
`threshold`	Fraction of the local trend below which an epoch is candidate rest (default 0.15).
`min_seed_period`	Minimum below-threshold run to seed a bout, in seconds (default 1800, 30 minutes).
`max_test_period`	Maximum bout length the consolidation tests, in seconds (default 43200, 12 hours).
`r_consec_below`	Above-threshold tolerance during consolidation, in seconds (default 1800, 30 minutes).
`nap_max_minutes`	A non-main bout shorter than this is labelled a nap (default 180).

Value

An object of class `actiRhythm_roenneberg`: a `rest_periods` data frame (one row per consolidated bout, with onset, offset, duration, and a main/nap label), the per-epoch `rest` vector, the activity `trend`, and per-bout counts. The function never errors; with no resolvable bout it returns an empty `rest_periods`.

References

Roenneberg T, Keller LK, Fischer D, Matera JL, Vetter C, Winnebeck EC (2015). "Human activity and rest in situ." *Methods in Enzymology*, **552**, 257–283. doi:10.1016/bs.mie.2014.11.028.

Loock A, Khan Sullivan A, Reis C, Paiva T, Ghotbi N, Pilz LK, Biller AM, Molenda C, Vuori-Brodowski MT, Roenneberg T, Winnebeck EC (2021). "Validation of the Munich Actimetry Sleep Detection Algorithm for estimating sleep-wake patterns from activity recordings." *Journal of Sleep Research*, **30**(6), e13371. doi:10.1111/jsr.13371.

Hammad G, Reyt M, Beliy N, Baillet M, Deantoni M, Lesoinne A, Muto V, Schmidt C (2021). "pyActigraphy: open-source python package for actigraphy data visualization and analysis." *PLOS Computational Biology*, **17**(10), e1009514. doi:10.1371/journal.pcbi.1009514.

See Also

sleep.changepoints, sleep.cole.kripke

Examples

```
# Three nights of rest plus one daytime nap
ts <- seq(as.POSIXct("2024-01-01", tz = "UTC"), by = 60, length.out = 3 * 1440)
h  <- as.numeric(format(ts, "%H")) + as.numeric(format(ts, "%M")) / 60
day <- as.integer(format(ts, "%j"))
counts <- ifelse(h >= 23 | h < 7, 5, 300)
counts[h >= 14 & h < 15.5 & day == min(day) + 1L] <- 5   # a nap on day 2
rest.periods(counts, ts)
```

---

rest.spt *Sleep-Period-Time Window from the z-Angle (HDCZA)*

Description

Detects the main sleep-period-time (SPT) window per day directly from the z-angle, with no sleep diary, using the van Hees et al. (2018) Heuristic algorithm based on the Distribution of Change in Z-Angle (HDCZA): the absolute 5-second change in arm angle is smoothed over 5 minutes, thresholded at a fraction of its own distribution, and the longest sustained low-change block per noon-to-noon day becomes the SPT. This is raw-native: there is no count-based equivalent.

Usage

```
rest.spt(
  anglez,
  timestamps,
  epoch_length = 5,
  pct = 10,
  mult = 15,
  clamp = c(0.13, 0.5),
  min_block = 30,
  max_gap = 60,
  algo = c("HDCZA", "HorAngle"),
  wear = NULL
)
```

Arguments

anglez	Numeric z-angle (degrees) per epoch, e.g. from raw.metrics(..., metrics = "anglez") at a short epoch.
timestamps	POSIXct timestamps, one per epoch.
epoch_length	Epoch length in seconds (default 5, the validated value).
pct	Percentile of the change distribution for the threshold (default 10).
mult	Multiplier on that percentile (default 15).
clamp	Lower/upper clamp (degrees) on the threshold (default c(0.13, 0.50)).
min_block	Minimum SPT block length in minutes (default 30).
max_gap	Maximum movement gap to bridge within the SPT, minutes (default 60).
algo	"HDCZA" (wrist, change-in-angle) or "HorAngle" (a hip variant thresholding the absolute angle at 60 degrees).
wear	Optional logical wear mask, one per epoch (e.g. from detect.nonwear.raw); non-wear epochs are excluded from the SPT so a stationary, taken-off device is not scored as sleep.

Value

An object of class actiRhythm_spt: a data frame with one row per day (date, onset, offset, duration in hours, and the threshold used). Never errors; returns no rows if no day has a detectable window.

References

van Hees VT, Sabia S, Jones SE, Wood AR, Anderson KN, Kivimaki M, Frayling TM, Pack AI, Bucan M, Trenell MI, Mazzotti DR, Gehrman PR, Singh-Manoux BA, Weedon MN (2018). "Estimating sleep parameters using an accelerometer without sleep diary." *Scientific Reports*, **8**, 12975. doi:10.1038/s4159801831266z.

See Also

sib.vanhees, sleep.from.spt, raw.metrics

Examples

```
# One still night inside an active day yields one SPT window
ts <- seq(as.POSIXct("2024-01-01 12:00", tz = "UTC"), by = 5, length.out = 17280)
h <- as.numeric(format(ts, "%H")); night <- h >= 23 | h < 7
set.seed(1)
anglez <- ifelse(night, -60, -30) + rnorm(17280, 0, ifelse(night, 0.02, 20))
rest.spt(anglez, ts, epoch_length = 5)
```

`rhythmicity.test` *Cosinor Rhythmicity Test*

Description

Tests whether a rhythm of the given period is present, using the Halberg zero-amplitude F-test (H0: amplitude = 0) on the single-component cosinor fit, and reports the percent rhythm (the proportion of variance the cosinor explains, R^2). It reuses `cosinor.analysis` as the single fitting engine, so the MESOR/amplitude/acrophase and valid-day gating are identical.

Usage

```
rhythmicity.test(
  counts,
  timestamps,
  period = 24,
  alpha = 0.05,
  wear_time = NULL,
  min_valid_hours = 10,
  cosinor_result = NULL
)
```

Arguments

`counts`	Numeric vector of activity counts (or any activity measure).
`timestamps`	POSIXct timestamps, one per count.
`period`	Rhythm period in hours (default 24).
`alpha`	Significance level for the `rhythmic` flag (default 0.05).
`wear_time`	Optional logical wear-time mask passed to the cosinor fit.
`min_valid_hours`	Minimum valid hours per day for the cosinor fit.
`cosinor_result`	Optional existing `cosinor.analysis` result to test without refitting; when supplied, `counts`/`timestamps` are ignored.

Value

An object of class `actiRhythm_rhythmicity`: a list with the F statistic, numerator/denominator degrees of freedom (`df1`, `df2`), `p_value`, `percent_rhythm` (and `r_squared`), a logical `rhythmic` flag, and the cosinor parameters.

References

Nelson W, Tong YL, Lee JK, Halberg F (1979). "Methods for cosinor-rhythmometry." *Chronobiologia*, **6**(4), 305–323.

Examples

```
ts <- seq(as.POSIXct("2024-01-01", tz = "UTC"), by = 60, length.out = 1440)
h <- as.numeric(format(ts, "%H"))
counts <- pmax(0, 100 + 80 * cos(2 * pi * (h - 14) / 24))
rhythmicity.test(counts, ts)
```

---

rolling_mean *Rolling Mean*

Description

Rolling Mean

Usage

```
rolling_mean(x, window)
```

Arguments

x	Numeric vector
window	Window size

Value

Rolling means

---

rolling_sd *Rolling Standard Deviation*

Description

Rolling Standard Deviation

Usage

```
rolling_sd(x, window)
```

Arguments

x	Numeric vector
window	Window size

Value

Rolling SDs

rolling_sum *Rolling Sum*

Description

Rolling Sum

Usage

```
rolling_sum(x, window)
```

Arguments

x	Numeric vector
window	Window size

Value

Rolling sums

save.circadian.plot *Save a actiRhythm Plot to a File*

Description

Convenience wrapper around ggplot2::ggsave for the plot_* circadian figures; the output format is taken from the file extension (.png, .pdf, .svg).

Usage

```
save.circadian.plot(plot, file, width = 8, height = 5, dpi = 300)
```

Arguments

plot	A ggplot object (e.g. from plot_actogram).
file	Output path; its extension sets the format.
width, height	Size in inches (defaults 8 x 5).
dpi	Resolution for raster formats (default 300).

Value

The file path, invisibly.

scale_color_actiRhythm *ggplot2 Scale for actiRhythm Colors*

Description

Discrete color scale using the actiRhythm palette.

Usage

```
scale_color_actiRhythm(type = "categorical", ...)
```

Arguments

type	Character. Palette type (see actiRhythm_colors)
...	Additional arguments passed to ggplot2::scale_color_manual

Value

A ggplot2 scale object

scale_fill_actiRhythm *ggplot2 Fill Scale for actiRhythm Colors*

Description

Discrete fill scale using the actiRhythm palette.

Usage

```
scale_fill_actiRhythm(type = "categorical", ...)
```

Arguments

type	Character. Palette type (see actiRhythm_colors)
...	Additional arguments passed to ggplot2::scale_fill_manual

Value

A ggplot2 scale object

set_actiRhythm_theme *Set actiRhythm Theme as Default*

Description

Sets theme_actiRhythm() as the default ggplot2 theme for the session.

Usage

```
set_actiRhythm_theme(...)
```

Arguments

...	Arguments passed to theme_actiRhythm()

Value

NULL, invisibly; called for its side effect of setting theme_actiRhythm() as the session default ggplot2 theme.

sib.vanhees *Sustained-Inactivity-Bout Sleep Scoring from the z-Angle*

Description

Scores each epoch sleep ("S") or wake ("W") from the z-angle by the van Hees et al. (2015) sustained-inactivity-bout rule: an interval with no arm-posture change exceeding angle_thresh degrees for at least time_thresh minutes is sustained inactivity (sleep). The output has the same shape as sleep.cole.kripke, so it feeds sleep.regularity.index, lids and the rest consumers directly. Intersect it with rest.spt via sleep.from.spt.

Usage

```
sib.vanhees(anglez, angle_thresh = 5, time_thresh = 5, epoch_length = 5)
```

Arguments

anglez	Numeric z-angle (degrees) per epoch.
angle_thresh	Posture-change threshold in degrees (default 5).
time_thresh	Minimum sustained-inactivity duration in minutes (default 5).
epoch_length	Epoch length in seconds (default 5).

Value

Character vector of "S"/"W", one per epoch. A day with fewer than 10 posture changes is scored all "S" (sustained inactivity, to be gated by the SPT window and wear time).

References

van Hees VT, Sabia S, Anderson KN, Denton SJ, Oliver J, Catt M, Abell JG, Kivimaki M, Trenell MI, Singh-Manoux A (2015). "A novel, open access method to assess sleep duration using a wrist-worn accelerometer." *PLoS ONE*, **10**(11), e0142533. doi:10.1371/journal.pone.0142533.

See Also

`rest.spt`, `sleep.from.spt`

Examples

```
# A long still stretch scores as sustained inactivity (sleep)
set.seed(1)
anglez <- c(rnorm(3000, -60, 0.02), rnorm(3000, 0, 20))   # still, then active
table(sib.vanhees(anglez, epoch_length = 5))
```

`sleep.changepoints` *Change-Point Detection of Sleep and Wake Onsets*

Description

Locates the sleep-onset and wake-onset time of each circadian cycle with a cosinor-anchored mean-shift change point. A fixed 24-hour cosinor bounds each rest and active span roughly (the cosinor anchoring follows CircaCP, Chen and Sun 2024), and the precise transition inside each bound is then placed with a single least-squares mean-shift change point on the raw counts. The result is a per-night sleep-onset / wake-onset table. A single rest-activity transition rate (such as `state.transitions`) cannot localise this timing.

Usage

```
sleep.changepoints(
  counts,
  timestamps,
  period = 1440,
  thr = 0.2,
  window_minutes = 240
)
```

Arguments

`counts`	Numeric activity vector (minute epochs recommended).
`timestamps`	POSIXct timestamps, one per value.
`period`	Cosinor period in minutes (default 1440, one day).
`thr`	Dichotomisation threshold on the range-scaled cosine, in $[0, 1]$ (default 0.2, approximating CircaCP's lower-20\ fitted curve above `thr` is the rough active span, below it the rough rest span.

`window_minutes` Half-width of the search window, in minutes, in which each rough boundary is refined to a change point (default 240).

Value

An object of class `actiRhythm_changepoints`: the cosinor summary, a `changepoints` data frame (time and type, "sleep onset" or "wake onset"), a `sleep_episodes` data frame (sleep onset, wake onset, and duration in hours), and the mean sleep duration. The function never errors; on insufficient data it returns the structure with `insufficient = TRUE`.

References

Chen S, Sun X (2024). "Validating CircaCP: a generic sleep-wake cycle detection algorithm for unlabelled actigraphy data." *Royal Society Open Science*, **11**(5), 231468. doi:10.1098/rsos.231468.

See Also

`state.transitions`, `sleep.regularity.index`

Examples

```
# Five days with a clear active day / restful night
ts <- seq(as.POSIXct("2024-01-01", tz = "UTC"), by = 60, length.out = 5 * 1440)
h  <- as.numeric(format(ts, "%H"))
counts <- ifelse(h >= 8 & h < 23, 300, 5) + pmax(0, stats::rnorm(length(ts), 0, 5))
sleep.changepoints(counts, ts)
```

`sleep.cole.kripke` *Cole-Kripke Sleep/Wake Scoring*

Description

Classifies each epoch as sleep or wake from activity counts with the Cole-Kripke algorithm (Cole et al. 1992), validated for adults on one-minute epochs, with the optional Webster rescoring rules (Webster et al. 1982). Returns the per-epoch `sleep_state` that the regularity and locomotor-sleep metrics read (`sleep.regularity.index`, `lids`).

Usage

```
sleep.cole.kripke(
  counts,
  apply_rescoring = TRUE,
  epoch_length = 60,
  na_action = c("na", "wake", "zero")
)
```

Arguments

counts	Numeric vector of activity counts (vertical axis).
apply_rescoring	Apply Webster's rescoring rules (default TRUE).
epoch_length	Epoch length in seconds (default 60). The algorithm was validated on 60-second epochs; other lengths raise a warning.
na_action	How NA-count epochs appear in the output: "na" (default) emits NA, so non-wear gaps are not read as sleep; "wake" scores them wake; "zero" scores them from a zero count.

Details

The sleep index uses a seven-epoch window (four before, the current epoch, and two after):

$$D = 0.001(106P_4 + 54P_3 + 58P_2 + 76P_1 + 230C + 74N_1 + 67N_2)$$

These are Cole's mean-activity-per-minute weights (Table 6, p.466); the $/100$ count scaling and the 300 cap are implementation conventions. An epoch is scored sleep when $D < 1$. Webster's rescoring then re-labels short sleep bouts that follow or are surrounded by sustained wake as wake.

Value

Character vector of states, "S" (sleep) or "W" (wake), the same length as counts.

References

Cole RJ, Kripke DF, Gruen W, Mullaney DJ, Gillin JC (1992). "Automatic sleep/wake identification from wrist activity." *Sleep*, **15**(5), 461–469. doi:10.1093/sleep/15.5.461.

Webster JB, Kripke DF, Messin S, Mullaney DJ, Wyborney G (1982). "An activity based sleep monitor system for ambulatory use." *Sleep*, **5**(4), 389–399. doi:10.1093/sleep/5.4.389.

See Also

sleep.sadeh, sleep.regularity.index

Examples

```
agd <- agd.counts(read.agd(example_agd(1), verbose = FALSE))
state <- sleep.cole.kripke(agd$axis1)
table(state)
```

`sleep.from.spt` *Sleep Parameters from SPT and Sustained-Inactivity Bouts*

Description

Intersects the per-epoch sustained-inactivity sleep score (`sib.vanhees`) with the sleep-period-time window (`rest.spt`) to derive per-night sleep parameters: total sleep time, onset and wake, wake-after-sleep-onset, efficiency, and awakenings.

Usage

```
sleep.from.spt(spt, sib, timestamps, epoch_length = 5)
```

Arguments

`spt`	An `actiRhythm_spt` object from `rest.spt`.
`sib`	A character `"S"/"W"` vector from `sib.vanhees`.
`timestamps`	POSIXct timestamps, one per epoch (matching `sib`).
`epoch_length`	Epoch length in seconds (default 5).

Value

An object of class `actiRhythm_sleep`: a data frame with one row per night (`date`, sleep `onset`, `offset`, `tst` hours, `waso` minutes, `efficiency` as a 0-1 fraction (TST/SPT), `n_awakenings`, `mid_sleep`).

References

van Hees VT, Sabia S, Jones SE, Wood AR, Anderson KN, Kivimaki M, Frayling TM, Pack AI, Bucan M, Trenell MI, Mazzotti DR, Gehrman PR, Singh-Manoux BA, Weedon MN (2018). "Estimating sleep parameters using an accelerometer without sleep diary." *Scientific Reports*, **8**, 12975. doi:10.1038/s4159801831266z.

See Also

`rest.spt`, `sib.vanhees`

Examples

```
ts <- seq(as.POSIXct("2024-01-01 12:00", tz = "UTC"), by = 5, length.out = 17280)
h <- as.numeric(format(ts, "%H")); night <- h >= 23 | h < 7
set.seed(1)
anglez <- ifelse(night, -60, -30) + rnorm(17280, 0, ifelse(night, 0.02, 20))
spt <- rest.spt(anglez, ts, epoch_length = 5)
sib <- sib.vanhees(anglez, epoch_length = 5)
sleep.from.spt(spt, sib, ts, epoch_length = 5)
```

sleep.regularity.index *Calculate Sleep Regularity Index (SRI) - Exported Version*

Description

Calculate Sleep Regularity Index (SRI) - Exported Version

Usage

```
sleep.regularity.index(sleep_state, timestamps, epoch_length = 60)
```

Arguments

sleep_state	Character vector of sleep states ("S" or "W")
timestamps	POSIXct timestamps (must be regular epochs)
epoch_length	Epoch length in seconds (default 60)

Value

Numeric SRI value (-100 to 100)

References

Phillips AJK, Clerx WM, O'Brien CS, Sano A, Barger LK, Picard RW, Lockley SW, Klerman EB, Czeisler CA (2017). "Irregular sleep/wake patterns are associated with poorer academic performance and delayed circadian and sleep/wake timing." *Scientific Reports*, **7**(1), 3216. doi:10.1038/s41598017031714.

sleep.sadeh *Sadeh Sleep/Wake Scoring*

Description

Classifies each epoch as sleep or wake from activity counts with the Sadeh algorithm (Sadeh et al. 1994), validated on adults and adolescents on one-minute epochs.

Usage

```
sleep.sadeh(
  counts,
  epoch_length = 60,
  wake_threshold = 0,
  clip = NULL,
  na_action = c("na", "wake", "zero")
)
```

Arguments

counts	Numeric vector of activity counts (vertical axis).
epoch_length	Epoch length in seconds (default 60). The algorithm was validated on 60-second epochs; other lengths raise a warning.
wake_threshold	Sleep-index cut: an epoch is sleep when $SI \geq$ wake_threshold (default 0, Sadeh 1994; ActiLife uses -4).
clip	Optional upper cap on counts before scoring (default NULL, no cap; ActiLife and pyActigraphy use 300). Affects AVG, SD, and LG.
na_action	How NA-count epochs appear in the output: "na" (default) emits NA, so non-wear gaps are not read as sleep; "wake" scores them wake; "zero" scores them from a zero count.

Details

The sleep index uses an eleven-epoch window (five before, the current epoch, and five after):

$$SI = 7.601 - 0.065 \cdot AVG - 1.08 \cdot NATS - 0.056 \cdot SD - 0.703 \cdot LG$$

where AVG is the window mean, $NATS$ the number of epochs with counts in [50, 100), SD the standard deviation over the current and five preceding epochs, and $LG = \log(\text{count} + 1)$. An epoch is scored sleep when $SI \geq 0$ (Sadeh et al. 1994). For ActiLife/ActiGraph parity, set wake_threshold = -4 and clip = 300.

Value

Character vector of states, "S" (sleep) or "W" (wake), the same length as counts.

References

Sadeh A, Sharkey KM, Carskadon MA (1994). "Activity-based sleep-wake identification: an empirical test of methodological issues." *Sleep*, **17**(3), 201–207. doi:10.1093/sleep/17.3.201.

See Also

sleep.cole.kripke, sleep.regularity.index

Examples

```
agd <- agd.counts(read.agd(example_agd(1), verbose = FALSE))
state <- sleep.sadeh(agd$axis1)
table(state)
```

`social.jet.lag` *Calculate Social Jet Lag*

Description

Difference between sleep midpoint on work days and free days.

Usage

```
social.jet.lag(sleep_periods, work_days = NULL)
```

Arguments

`sleep_periods`	Data frame with in_bed_time, out_bed_time columns
`work_days`	Optional. Logical vector, date vector, or NULL (uses Mon-Fri default)

Details

Social jet lag is the discrepancy between social and biological time, computed as MSF - MSW (midpoint on free days minus work days).

`MSFsc` applies the MCTQ sleep-debt correction (Roenneberg et al. 2012): when free-day sleep exceeds work-day sleep (sleep catch-up), the free-day mid-sleep is adjusted down by half the excess over the weekly average sleep duration, and `SJLsc` = `MSFsc` − `MSW`. These are based on time in bed (the supplied in-bed/out-bed times), not on sleep onset and offset.

Positive values (most common) indicate later sleep timing on free days. Values > 1 hour are associated with increased health risks.

Value

List with social jet lag metrics, including `MSW`/`MSF` (mid-sleep on work and free days), `social_jet_lag_hours` (MSF - MSW), the sleep-debt-corrected chronotype `MSFsc` and corrected `social_jet_lag_sc_hours` (SJLsc), and the number of work/free nights.

References

Wittmann M, Dinich J, Merrow M, Roenneberg T (2006). "Social jetlag: misalignment of biological and social time." *Chronobiology International*, **23**(1-2), 497–509. doi:10.1080/07420520500545979.

Roenneberg T, Allebrandt KV, Merrow M, Vetter C (2012). "Social jetlag and obesity." *Current Biology*, **22**(10), 939–943. doi:10.1016/j.cub.2012.03.038.

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

`sri.matrix` *Sleep Regularity Index (SRI) - Phillips (2017) Epoch-of-Day x Day Matrix*

Description

Computes the Sleep Regularity Index exactly as defined by Phillips et al. (2017) using the full epoch-of-day by day concordance matrix. This is the published form of the SRI. Unlike a single fixed 24-hour-lag comparison, every epoch is binned by its real clock time (epoch-of-day) and by calendar date, a binary sleep/wake matrix is built, and the index is the average agreement between the SAME clock time on CONSECUTIVE calendar days, rescaled to the interval [-100, 100].

Usage

```
sri.matrix(sleep_state, timestamps, epoch_length = 60)
```

Arguments

`sleep_state`	Sleep/wake state per epoch. Either a character vector of `"S"` (sleep) / `"W"` (wake), or a numeric/integer/logical vector where 1/TRUE = sleep and 0/FALSE = wake. `NA` marks unscored / non-wear epochs and is excluded from the concordance count. Must be the same length as `timestamps`.
`timestamps`	POSIXct vector of epoch start times, one per element of `sleep_state`. Need not start at midnight and need not be perfectly contiguous; alignment is by real clock time.
`epoch_length`	Epoch length in seconds (default 60). Must divide 86400 evenly; `M = 86400 / epoch_length` is the number of epochs per day.

Details

Let `M = 86400 / epoch_length` be the number of epochs in a day and let `N` be the number of calendar days spanned by the recording. Each epoch is assigned an epoch-of-day index `i` (1..M) from its clock time (hour/minute/second) and a day index `j` (1..N) from its calendar date. These two indices populate a binary matrix `s[i, j]` where 1 = sleep and 0 = wake. The SRI is then

$$SRI = -100 + \frac{200}{M(N-1)} \sum_{i=1}^{M} \sum_{j=1}^{N-1} \mathbf{1}\{s_{i,j} = s_{i,j+1}\}$$

i.e. the proportion of clock-time epochs that hold the same state on two consecutive days, mapped linearly so that perfect regularity scores +100, chance-level (independent) scoring about 0, and perfect anti-regularity -100.

Robustness to gaps and partial wear: any consecutive-day pair `(i, j) -> (i, j + 1)` in which either cell is `NA` (missing epoch, non-wear, or a calendar day not represented in the data) is skipped and NOT counted. The sum is divided by the actual number of valid pairs rather than the theoretical `M * (N - 1)`, so missing data reduce statistical power but do not bias the estimate.

Days are aligned by true clock time using `as.POSIXlt()` (hour, minute, second give the epoch-of-day; the calendar date gives the day index), so the result is correct even when a recording does not start at midnight.

A `SRI` of `NA_real_` is returned (never an error) when there are fewer than two days, no valid consecutive-day pairs, or no usable input. The return structure is always the same list.

Value

A list with components:

SRI Numeric Sleep Regularity Index in [-100, 100] (rounded to 2 dp), or `NA_real_` if it cannot be computed.

n_days Integer number of distinct calendar days (N) spanned.

n_valid_pairs Integer number of consecutive-day epoch pairs that were actually compared (both cells non-NA).

method Character constant `"phillips_matrix"`.

References

Phillips AJK, Clerx WM, O'Brien CS, Sano A, Barger LK, Picard RW, Lockley SW, Klerman EB, Czeisler CA (2017). "Irregular sleep/wake patterns are associated with poorer academic performance and delayed circadian and sleep/wake timing." *Scientific Reports*, **7**(1), 3216. doi:10.1038/s41598017031714.

Examples

```
# A perfectly regular sleeper: same 8h sleep block every day -> SRI near 100
ts <- seq(as.POSIXct("2024-01-01 00:00:00", tz = "UTC"),
          by = 60, length.out = 1440 * 4)
tod <- as.POSIXlt(ts)$hour
state <- ifelse(tod < 8, "S", "W")   # asleep 00:00-08:00 every day
sri.matrix(state, ts, epoch_length = 60)$SRI
```

`state.transitions` *Rest-Activity State Transition Rates (kRA, kAR)*

Description

Computes the rest-to-activity and activity-to-rest transition rates from a binarized activity series, following the survival-curve method used by pyActigraphy (Lim et al. 2011). Thresholds the series into rest/active epochs, builds the per-lag transition probability (hazard) of each bout type from the bout-length survival curve, and takes a single rate over the LOWESS "sustained" plateau of that curve.

Usage

```
state.transitions(counts, threshold = 1, frac = 0.3, iter = 0)
```

Arguments

counts	Numeric activity vector; NA are dropped.
threshold	Activity level at or above which an epoch is "active" (default 1, i.e. any non-zero count).
frac	LOWESS smoother span for the sustained-region search (default 0.3).
iter	LOWESS robustifying iterations (default 0).

Value

An object of class actiRhythm_transitions: a list with kRA/kAR (sustained rest-to-active / active-to-rest rates over the LOWESS plateau), pRA/pAR (overall rest-to-active / active-to-rest rate, the reciprocal mean bout length), bout counts, and the two transition curves. The plateau search follows the pyActigraphy implementation of Lim et al. (2011).

References

Lim ASP, Yu L, Costa MD, Buchman AS, Bennett DA, Leurgans SE, Saper CB (2011). "Quantification of the fragmentation of rest-activity patterns in elderly individuals using a state transition analysis." *Sleep*, **34**(11), 1569–1581. doi:10.5665/sleep.1400.

Hammad G, Reyt M, Beliy N, Baillet M, Deantoni M, Lesoinne A, Muto V, Schmidt C (2021). "pyActigraphy: open-source python package for actigraphy data visualization and analysis." *PLOS Computational Biology*, **17**(10), e1009514. doi:10.1371/journal.pcbi.1009514.

Examples

```
set.seed(1)
counts <- as.integer(stats::runif(5000) < 0.1) * 100
state.transitions(counts)
```

---

theme_actiRhythm *actiRhythm ggplot2 Theme*

Description

A clean, professional theme for actiRhythm visualizations. Publication-ready with excellent readability and consistent with the dashboard's 6-size typography system.

Usage

```
theme_actiRhythm(base_size = 14, base_family = "", grid = TRUE, dark = FALSE)
```

Arguments

base_size	Numeric. Base font size in points (default: 14)
base_family	Character. Base font family (default: "")
grid	Logical. Show major grid lines? (default: TRUE)
dark	Logical. Use dark mode? (default: FALSE)

Details

Typography sizes (matching CSS):

- Caption/footnotes: 11px (0.79x base)
- Labels/metadata: 12px (0.86x base)
- Body text: 14px (base_size)
- Emphasis/titles: 16px (1.14x base)
- Section headings: 20px (1.43x base)
- Page titles: 24px (1.71x base)

Value

A ggplot2 theme object

Examples

```
library(ggplot2)
ggplot(mtcars, aes(wt, mpg)) +
  geom_point() +
  theme_actiRhythm()
```

---

transition.probability

Rest-Active Transition Probabilities

Description

Maximum-likelihood and Bayesian estimates of the rest-to-active and active-to-rest transition probabilities (Danilevicz et al. 2024): the transitions out of a state divided by its epochs at risk.

Usage

```
transition.probability(counts, threshold = 1, eps = 0.5)
```

Arguments

counts	Numeric activity vector; NA are dropped.
threshold	Counts at or above which an epoch is active (default 1).
eps	Bayesian Beta pseudo-count added to the transition count and the epochs at risk (default 0.5).

Value

A list with the maximum-likelihood and Bayesian tp_ra (rest to active) and tp_ar (active to rest), the active bout count, and the mean active bout length.

References

Danilevicz IM, van Hees VT, van der Heide F, Jacob L, Landre B, Benadjaoud MA, Sabia S (2024). "Measures of fragmentation of rest activity patterns: mathematical properties and interpretability based on accelerometer real life data." *BMC Medical Research Methodology*, **24**, 132. doi:10.1186/s1287402402255w.

See Also

`state.transitions`, `activity.balance.index`

Examples

```
counts <- c(rep(0, 50), rep(100, 20), rep(0, 40), rep(80, 30), rep(0, 60))
transition.probability(counts)
```

`ultradian.bandpower` *Ultradian Wavelet Band Power*

Description

Partitions the activity variance into dyadic period bands with an undecimated (MODWT) Haar wavelet transform, isolating ultradian bands such as the about-90-minute, about-4-hour, and about-8-hour rhythms (Percival and Walden 2000; Leise 2013). Reports the energy, power, and fraction of total variance in each band.

Usage

```
ultradian.bandpower(
  counts,
  timestamps,
  bands = list(`90min` = c(1, 2), `4h` = c(2, 6), `8h` = c(6, 12)),
  epoch_length = 60
)
```

Arguments

`counts`	Numeric activity vector (minute epochs recommended).
`timestamps`	POSIXct timestamps, one per value.
`bands`	Named list of `c(low_hours, high_hours)` period bands (default about 90 min, about 4 h, about 8 h).
`epoch_length`	Epoch length in seconds (default 60).

Value

An object of class `actiRhythm_bandpower`: a per-band table (energy, power, fraction) and the per-level wavelet variance. Never errors.

References

Percival DB, Walden AT (2000). *Wavelet Methods for Time Series Analysis*. Cambridge University Press. doi:10.1017/CBO9780511841040.

Leise TL (2013). "Wavelet analysis of circadian and ultradian behavioral rhythms." *Journal of Circadian Rhythms*, **11**, 5. doi:10.1186/17403391115.

Examples

```
ts <- seq(as.POSIXct("2024-01-01", tz = "UTC"), by = 60, length.out = 2 * 1440)
th <- as.numeric(difftime(ts, ts[1], units = "hours"))
ultradian.bandpower(100 + 50 * sin(2 * pi * th / 4), ts)   # a 4-hour rhythm
```

---

wavelet.coi *Wavelet Cone of Influence*

Description

The Morlet cone of influence (Torrence and Compo 1998): the largest period, in hours, at each time point below which the wavelet power is free of edge effects. Power at periods above this curve, near the start and end of the recording, is unreliable. `circadian.wavelet` returns this curve in its result; this helper computes it directly for plotting or masking.

Usage

```
wavelet.coi(timestamps, omega0 = 6, epoch_length = 60)
```

Arguments

`timestamps`	POSIXct timestamps, one per epoch.
`omega0`	Morlet central frequency (default 6).
`epoch_length`	Epoch length in seconds (default 60).

Value

Numeric vector, one per epoch, of the maximum reliable period (hours).

References

Torrence C, Compo GP (1998). "A practical guide to wavelet analysis." *Bulletin of the American Meteorological Society*, **79**(1), 61–78. doi:10.1175/15200477(1998)079<0061:APGTWA>2.0.CO;2.

See Also

`circadian.wavelet`

Examples

```
ts <- seq(as.POSIXct("2024-01-01", tz = "UTC"), by = 600, length.out = 6 * 144)
range(wavelet.coi(ts, epoch_length = 600))
```

`write.agd` *Write Activity Counts to an .agd File*

Description

Writes a per-epoch counts data frame to a SQLite `.agd` file (the standard `settings` + `data` schema), readable by `read.agd`.

Usage

```
write.agd(
  data,
  path,
  epoch_length = 60,
  start_time = NULL,
  device_serial = "",
  device_name = "wGT3XBT",
  subject_name = "",
  sample_rate = 30,
  mode = 61
)
```

Arguments

`data`	Data frame with `timestamp` (POSIXct) and `axis1`, `axis2`, `axis3`; optional `steps`, `lux` and `inclineOff`/`Standing`/`Sitting`/`Lying`.
`path`	Output `.agd` path.
`epoch_length`	Epoch length in seconds (default 60).
`start_time`	Recording start (POSIXct); defaults to the first timestamp.
`device_serial`, `device_name`, `subject_name`, `sample_rate`, `mode`	Metadata for the settings table.

Value

The output `path`, invisibly.

Index