

Package: ActiGlobe (via r-universe)

September 17, 2026

Type Package

Title Wearable Recording Processor for Time Shift Adjustment and Data Analysis

Version 0.2.2

Maintainer C. William Yao <chun.william.yao.cnmt1@ssss.gouv.qc.ca>

Description Cross-continental travel and local practices of daylight saving time can induce time shift. When left unadjusted, it can lead to biases in interpreting wearable measures. This research tool streamlines data processing and analysis (e.g., cosinor analysis, circularized kernel density estimation) for longitudinal recordings. Together with additional features such as exporting daily wearable measures and creating visual reports, it aims to facilitate research on longitudinal wearable measurements. To learn its application, please refer to the provided vignettes for a brief tutorial. This tool is developed based on the methodology detailed in: Varesco, G. Yao, C. W., Dubé, E., Simonelli, G., Bieuzen, F., (2024) <doi:10.1113/ep092195>.

License AGPL (>= 3)

Encoding UTF-8

LazyData true

RoxygenNote 7.3.3

Depends R (>= 4.2.0)

Imports ggplot2, ggrepel, grDevices, gridExtra, lubridate, readr, sandwich, utils, viridis, parallel, scales

Suggests spelling, ggthemes, knitr, rmarkdown, testthat (>= 3.0.0), zeallot, hms, mockery

URL <https://github.com/cwy20030/ActiGlobe>,
<https://cwy20030.github.io/ActiGlobe/>

BugReports <https://github.com/cwy20030/ActiGlobe/issues>

ByteCompile true

Config/testthat/edition 3
NeedsCompilation no
VignetteBuilder knitr
Language en-US
Config/pak/sysreqs libx11-dev
Repository <https://ropensci-staging.r-universe.dev>
Date/Publication 2026-02-04 22:57:10 UTC
RemoteUrl <https://github.com/cwy20030/ActiGlobe>
RemoteRef HEAD
RemoteSha 430c36a8ba48f812c30ae1fe562b57c063a936a0

Contents

Act2Daily	3
boot.seci	5
BriefSum	7
CosinorM	8
CosinorM.KDE	12
Date2TotalT	16
DateFormat	17
DST	19
DST2GL	20
FlyEast	21
FlyEast_adj	21
ggActiGlobe	22
ggCosinorM	23
IANA	25
Num2UTC	26
Rad2Hr	27
TAdjust	28
TimeFormat	29
TLog	30
TravelLog	31
TZ2UTC	31
UTC2Num	32
UTCwDST	33
write.act	34
write.cosinor	36

Description

‘Act2Daily’ is used to split the actigraphy recording by the recording dates stored in Bdf (a BriefSum object).

Usage

```
Act2Daily(
  df,
  Bdf,
  TUnit = "hour",
  VAct = NULL,
  VTm = NULL,
  Incomplete = FALSE,
  Travel = TRUE
)
```

Arguments

df	A data.frame of raw actigraphy recording. Both time and activity count should be included in the df. See VAct and VTm for further detail.
Bdf	A BriefSum object containing per-day metadata for the recording. Note, if jet lag occurred during the recording, please, update the metadata using TAdjust before passing to this function.
TUnit	Character; time-unit for the x-axis of each day’s timeline. Must be one of "day", "hour", "minute" or "second". Default is "hour".
VAct	Optional character. Name of the activity column in df. If NULL, defaults to the second column of df.
VTm	Optional character. Name of the date.time index column in df. If NULL, defaults to the first column of df.
Incomplete	Logical; if TRUE, days flagged ‘Incomplete Recording’ (i.e. <24 h) are retained in the data list with recordings segmented by day. Default = FALSE (these days are removed).
Travel	Logical; if TRUE, days flagged ‘Travel’ are retained although some data points from an earlier adjacent calendar day may be duplicated (a warning is issued). If FALSE, travel days and the day before/after are excluded. Default = TRUE.

Details

Transforms a continuous actigraphy time-series into a day-by-day list of aligned epoch and produces a fully annotated data frame. Based on the summary table generated by either [BriefSum](#) or [TAdjust](#), it re-organize and segment the raw recording data by each day. The function is designed

to process incomplete recordings while accounting for travel and daylight-saving adjustments. All labels contained in the summary table (i.e., Bdf) will be assigned to the corresponding data points. This workflow ensures that each day's recording is bounded by its calendar date while accounts for travel-induced overlaps, and clearly documents any incomplete or unallocated points.

Value

A data list that includes the adjusted recordings both in long format and as a nested list by day.

Note

When you cross into a time zone with a larger UTC offset (e.g., Montreal $UTC - 5$ to Taipei $UTC + 8$), it will generate a small overlap of epochs between the day before travel and the day labeled "Travel Day" in the "Daily_df". Conversely, when you move into a time zone with a smaller UTC offset, a brief gap appears on "Travel Day", with labeling - "Unallocated", which represents "time lose".

See Also

[TAdjust write.act](#)

Examples

```
## Not run:

# Import data
data (FlyEast)

# Create quick summary of the recording with adjustment for daylight saving.
BdfList <-
  BriefSum (
    df = FlyEast,
    SR = 1 / 60,
    Start = "2017-10-24 13:45:00"
  )

# Lets extract the quick summary of the recording
Bdf <- BdfList$Bdf

### Note that since the original data was affected by travel-induced time
### shift, the recordings would not be properly segmented from 2017-11-02.

## To avoid time shift due to travelling, we will keep only the first 8 days.
Bdf <- Bdf [1:8, ]
df <- BdfList$df

# Segment Data by Day
dfList <-
  Act2Daily (
    df = df,
```

```

        Bdf = Bdf,
        VAct = "Activity",
        VTm = "Time",
        Incomplete = TRUE,
        Travel = TRUE
    )

str(dfList) ### Look at the output structure
View(dfList)

## End(Not run)

```

boot.seci

*Bootstrap Standard Errors and Confidence Intervals for Parameters***Description**

Computes standard errors and confidence intervals for cosinor and post-hoc parameters via non-parametric bootstrap

Usage

```
boot.seci(object, ci_level = 0.95, n = 500, digits = 2)
```

Arguments

object	A fitted ‘CosinorM’ or ‘CosinorM.KDE’ model object.
ci_level	Numeric scalar. The threshold for the confidence interval. Default: 0.95
n	Numeric scalar. Numbers of bootstraps required to estimate the standard errors and confidence intervals. Default: 500
digits	Numeric scalar. Integer indicating the number of decimal places (round) to be used. Default: 2

Value

A data.frame with one row per cosinor coefficient and columns:

- Estimate: Mean of bootstrap coefficient values.
- Std Error: Bootstrap standard deviation of each coefficient across n resamples.
- t value: Ratio of the observed estimate to its bootstrap standard error, analogous to a signal-to-noise measure:

$$t = \hat{\theta}_{obs} / SE_{boot}$$

- lower CI label: Percentile lower bound at $\frac{\alpha}{2}$, where $\alpha = 1 - ci_{level}$.
- upper CI label: Percentile upper bound at $1 - \frac{\alpha}{2}$.

See Also[boot](#)**Examples**

```
## Not run:
# Import data
FlyEast

BdfList <-
  BriefSum (
    df = FlyEast,
    SR = 1 / 60,
    Start = "2017-10-24 13:45:00"
  )

# Let's extract actigraphy data from a single day
df <- BdfList$df
df <- subset (df, df$Date == "2017-10-27")

# Multicomponent Cosinor Model
fit <- CosinorM (
  time = df$Time,
  activity = df$Activity,
  tau = c (12, 24),
  method = "OLS"
)

# inspect coefficients

boot.seci (
  object = fit,
  ci_level = 0.95,
  n = 500
)

# Gaussian Kernel Density Estimation
fit2 <- CosinorM.KDE (
  time = df$Time,
  activity = df$Activity
)

# inspect coefficients

boot.seci (
  object = fit2,
  ci_level = 0.95,
  n = 500
)

## End(Not run)
```

BriefSum

*Summarize the Actigraphy Recording by Day***Description**

'BriefSum()' is a function that summarizes the actigraphy recording by day. It generates a data.frame where each row holds all metadata for one recording day: the calendar date, time-zone code, epoch length (seconds), UTC offset, daylight-saving flag, cumulative start/end seconds from midnight, recording start/end times (HH:MM:SS), any warning labels (e.g. 'Travel', 'Incomplete Recording'), an exclusion flag, and the expected number of epochs for a full day.

Usage

```
BriefSum(df, SR, Start, TZ = "local")
```

Arguments

df	A data.frame of raw actigraphy recording. Both time and activity count should be included in the df.
SR	The sampling rate of the actigraphy (unit at Hz). Note that Hz should be equal to or less than 1.
Start	The starting date and time of the recording in the format as "2021-03-05 18:31:03". See as.POSIXct for more details and DateFormat and TimeFormat for formatting help.
TZ	The time zone when the recording started. (default = "local", which means user's local time zone)

Value

A named list with two elements:

- Bdf A data.frame containing brief summary information of each recording day. Columns include:
 - Date calendar date (YYYY-MM-DD)
 - Epoch epoch length in seconds
 - UTC dominant UTC offset string (e.g., "UTC+02:00")
 - TZ_code time zone code (e.g., "EST")
 - Daylight_Saving logical flag indicating DST for the day
 - Recording_Start earliest recorded time for the day (HH:MM:SS)
 - Recording_End latest recorded time for the day (HH:MM:SS)
 - GL_Offset numeric offset returned by [DST2GL](#) for the day
 - nDataPoints number of epochs observed for the day
 - Cumulative_Start_Second cumulative start second from midnight for the day's first epoch

- Cumulative_End_Second cumulative end second from midnight for the day's last epoch
- Excluded logical flag; TRUE if the day is excluded (e.g., incomplete)
- Warning character; warning label when applicable (e.g., "Incomplete Recording", "Time Change")
- df The original input df augmented with additional columns (class ActiGlobe, data.frame):
 - DateTime POSIXct timestamp for each epoch (tz = TZ)
 - Date date string (YYYY-MM-DD)
 - Time time string (HH:MM:SS)
 - UTC UTC offset string for each epoch
 - DaylightSaving logical flag per epoch
 - nPoint cumulative epoch index

The function returns a list with both Bdf df = df) and sets classes c("ActiGlobe", "data.frame") on both returned data.frames.

Examples

```
# Import data
data (FlyEast)

BdfList <- BriefSum (
  df = FlyEast,
  SR = 1 / 60,
  Start = "2017-10-19 13:45:00"
)

str (BdfList)
# View(BdfList)

## install library "zeallot"
## library(zeallot)
## c(Bdf, df) %<-%
## BriefSum(df = FlyEast,
##          SR = 1/60,
##          Start = "2017-10-24 13:45:00")
```

Description

Fit a cosine-based harmonic linear regression on circular time (hours of day)

Usage

```
CosinorM(
  time,
  activity,
  tau,
  method = "OLS",
  arctan2 = TRUE,
  type = "HC3",
  dilute = FALSE
)
```

Arguments

time	Numeric vector of time coordinates for each data point
activity	Numeric vector of activity counts from an actigraphy device
tau	Numeric scalar or vector for the assumed circadian period. Default is 24 for single-phase; multiple phases can be supplied via <code>c()</code>
method	Character string specifying estimation method • "OLS": Ordinary least squares via lm (default) • "FGLS": Feasible generalized least squares; models heteroskedasticity via a log-variance fit to squared OLS residuals, computes weights, and refits by weighted least squares
arctan2	Logical; if TRUE (default) acrophase is computed with <code>atan2(gamma, beta)</code> , resulting in the quadrant interval between $-\pi$ and π . Whereas, when set to FALSE, the legacy arctangent quadrant is mapped. The resulting interval lies between $-\frac{\pi}{2}$ and $\frac{\pi}{2}$.
type	Character string passed to vcovHC for robust standard error computation
dilute	Logical; • "FALSE": All essential parameters would be produced. (default) • "TRUE": Only cosinor coefficients are returned. This is suited for post-hoc processes, such as computing confidence interval via nonparametric bootstrap

Details

The ‘Cosinor’ model is a cosine-based harmonic regression used to estimate circadian rhythm parameters.

Single-phase equation:

$$y = M + A \cos\left(\frac{2\pi t}{\tau} - \phi\right)$$

- M : MESOR (mid-line estimating statistic of rhythm), the intercept
- A : Amplitude, peak deviation from M
- t : Time coordinate within the cycle
- τ : The assumed period length

- ϕ : Acrophase (time-of-peak), computed from fitted sine and cosine coefficients

Note that because the model is parameterized with a negative phase inside the cosine, this means the the derived peak time from acrophase corresponds to the peak time forward from midnight (not backward). In other words, the when the acrophase is positive, it means the peak activity time occurs after midnight and vice versa.

Linearized form:

$$\hat{y} = M + \beta x + \gamma z + \epsilon$$

- $\beta = A * \cos(\phi)$, estimated coefficient for the cosine term
- $x = \cos(\frac{2*\pi*t}{\tau})$, cosine-transformed time
- $\gamma = A * \sin(\phi)$, estimated coefficient for the sine term
- $z = \sin(\frac{2*\pi*t}{\tau})$, sine-transformed time
- ϵ : error term

Model parameters are estimated by minimizing the residual sum of squares:

$$RSS = \sum_{i=1}^n (y_i - (M + \beta x_i + \gamma z_i))^2$$

By default, `lm` fits this via QR decomposition.

Acrophase ϕ interpretation: ϕ is derived from

$$\phi = \arctan 2\left(\frac{\gamma}{\beta}\right)$$

Note,

- ϕ can be converted to clock time to identify the peak activity time AFTER midnight.
- if the model is not a mono-phase (i.e., more than one τ), there will be $\frac{24(hours)}{\tau}$ number of peaks within a day.

Amplitude (A) estimation: Amplitude is calculated from fitted sine and cosine coefficients as:

$$A = \sqrt{(\beta^2 + \gamma^2)}$$

Value

A list of class `c("CosinorM", "lm")` containing:

- `parm`: Parameters specified in the model
- `tau`: The assumed period length
- `time`: The time coordinates of the recording
- `method`: The estimation method used
- `coef.cosinor`: Named numeric vector with entries:
 - `MESOR`: Mid-line estimating statistic of rhythm
 - `Amplitude`: Predicted peak activity magnitude
 - `Acrophase`: Acrophase in radian (time-of-peak)

- Beta: the coefficient of the cosine function
- Gamma: the coefficient of the sine function
- post.hoc: Post-hoc peak/trough diagnostics derived from fitted.values at observation angles (MESOR.ph, Bathyphase.ph.time, Trough.ph, Acrophase.ph.time, Peak.ph, Amplitude.ph)
- extra: only available for ultradian (i.e., τ less than 24hour) single-component model
- vcov: Robust variance-covariance matrix
- se: Standard errors

Inherits all components from [lm](#).

References

Cornelissen G. Cosinor-based rhythmometry. Theoretical Biology and Medical Modelling. 2014-12-01 2014;11(1):16. doi:10.1186/1742-4682-11-16

Chambers, J. M. (1992) Linear models. Chapter 4 of Statistical Models in S eds J. M. Chambers and T. J. Hastie, Wadsworth & Brooks/Cole.

Wilkinson, G. N. and Rogers, C. E. (1973). Symbolic descriptions of factorial models for analysis of variance. Applied Statistics, 22, 392-399. doi:10.2307/2346786.

Harvey, A. C. (1976). Estimating Regression Models with Multiplicative Heteroscedasticity. Econometrica, 44(3), 461-465. doi:10.2307/1913974

See Also

[lm](#) [CosinorM.KDE](#) [ggCosinorM](#)

Examples

```
require (stats)
require (graphics)

## Not run:
# Import data
data( FlyEast)

# Create quick summary of the recording with adjustment for daylight saving.
BdfList <-
  BriefSum (
    df = FlyEast,
    SR = 1 / 60,
    Start = "2017-10-24 13:45:00"
  )

# Let's extract actigraphy data from a single day
df <- BdfList$df
df <- subset (df, df$Date == "2017-10-27")

fit <- CosinorM (
  time = df$Time,
```

```

    activity = df$Activity,
    tau = 24,
    method = "OLS"
)

# inspect coefficients
fit$coef.cosinor

# plot Cosinor in hours
plot(fit$time,
     fit$fitted.values,
     type = "l",
     xlab = "Hour",
     ylab = "24-Hour Cosinor Model")

## End(Not run)

```

CosinorM.KDE

KDE-based circadian cosinor summary

Description

Fit an intensity-weighted Gaussian kernel density estimate (KDE) on circular time (hours of day) weighted by activity and extract first-harmonic cosinor summaries.

Usage

```
CosinorM.KDE(time, activity, bw = 0.8, arctan2 = TRUE, dilute = FALSE)
```

Arguments

time	Numeric vector of time coordinates for each data point. Values are interpreted modulo 24 and must lie in $[0, 24)$. If not numeric, an internal function will be used to convert it to numeric values in unit of hour.
activity	Numeric vector of activity counts from an actigraphy device in correspondance to each time point. Non-numeric inputs will be coerced. All-zero or non-finite activity values produce an error.
bw	Numeric scalar. Kernel standard deviation (SD) controlling s moothing. bw is interpreted in the same units as the angular scale after conversion: the implementation converts 'bw' from hours to radians internally using $bw_{rad} = bw * 2\pi/\tau$. Default: 0.8. Note, small bw (approaching 0) can cause numerical instability, while bw (approaching 1.5) leads a near-uniform kernel and reduces temporal resolution.

arctan2	Logical; if TRUE (default) acrophase is computed with $\text{atan2}(\text{gamma}, \text{beta})$, resulting in the quadrant interval between $-\pi$ and π . Whereas, when set to FALSE, the legacy arctangent quadrant is mapped. The resulting interval lies between $-\frac{\pi}{2}$ and $\frac{\pi}{2}$.
dilute	Logical; if FALSE (default), all essential parameters would be produced. When set to TRUE, only cosinor coefficients are returned. This is suited for post-hoc processes, such as computing confidence interval via nonparametric bootstrap

Details

This function builds a kernel-smoothed circular activity intensity from event time (hours in $[0, 24)$) and activity. The intensity is estimated on a regular angular grid over $[0, 2\pi)$ using a wrapped Gaussian kernel with specified bandwidth and normalized to form a circular density over phase. Cosinor summaries (MESOR, Beta, Gamma, Amplitude, Acrophase) are obtained by numerical integration of the fitted density using trapezoid-rule quadrature.

Cosinor parameters are obtained by numerical integration of the KDE on the grid:

- MESOR: mean level of the fitted curve, $M = I_0/(2\pi)$.
- β : cosine coefficient, $\beta = I_{\cos}/\pi$.
- γ : sine coefficient, $\gamma = I_{\sin}/\pi$.
- Amplitude: magnitude of the first harmonic, $A = \sqrt{\beta^2 + \gamma^2}$.
- Acrophase: peak time, $\phi = \text{atan2}(-\gamma, \beta)$, expressed in radians and converted to hours.

The grid integrals are computed with trapezoid weights w :

$$I_0 = \text{area} \sum \frac{\text{pdf } w}{\text{den}}$$

$$I_{\cos} = \text{area} \sum \frac{\text{pdf } \cos(\theta) w}{\text{den}}$$

$$I_{\sin} = \text{area} \sum \frac{\text{pdf } \sin(\theta) w}{\text{den}}$$

where pdf is the normalized KDE at grid angle θ , den is the kernel denominator at that grid point, and area rescales the fitted density back to the activity scale.

Notes:

- Normalization factors $1/\pi$ and $1/(2\pi)$ follow from the orthogonality of sine and cosine on $[0, 2\pi)$.
- The KDE summarizes the first harmonic of the rest-activity pattern over one cycle; higher harmonics are not extracted.
- The trapezoid rule ensures numerical stability by integrating over a dense, evenly spaced grid.
- The dense grid excludes the duplicate 2π endpoint (grid in $[0, 2\pi)$) to avoid duplicated quadrature nodes.
- Grid and observation trapezoid weights correct for irregular sampling and yield stable approximations to continuous integrals.

- Grid points with near-zero denominator $\sum_i K(\theta - \theta_i)w_i$ are set to NA and excluded from integrals to avoid division-by-zero artifacts.

Residual Variance and Effective Degree of Freedom

- **Regression (projection hat matrix):**

$$\hat{\sigma}^2 = \frac{\text{RSS}}{n - p},$$

where p is the number of parameters (degrees of freedom used by the fit).

- **Linear kernel smoothing:**

Fitted values are given by the linear smoother

$$\hat{y} = Wy,$$

where W is the hat matrix induced by the kernel weights (not a projection).

When defining residuals as $r = y - \hat{y}$ and $\text{RSS} = \sum r_i^2$, the unbiased estimator under the Gaussian kernel smoother can be written as

$$\hat{\sigma}^2 = \frac{\text{RSS}}{n - 2 \text{tr}(W) + \text{tr}(W^\top W)}.$$

where,

•

$$\text{tr}(W)$$

is the effective degrees of freedom used by the smoother (analogous to the number of parameters).

•

$$\text{tr}(W^\top W) = \sum_{i,j} W_{ij}^2$$

is a correction term because W is not an orthogonal projection.

Value

A list of class `c("CosinorM.KDE")` with elements:

- `parm`: Parameters specified in the model (list with `time`, `activity`, `bw`)
- `tau`: Period in hours (default to 24hour)
- `kdf`: A `data.frame` with the following columns:
 - `theta`: Angular positions (radians) at the observation angles
 - `density`: Estimated PDF values at the observation angles (integrates to 1 over $[0, 2\pi)$ when scaled)
 - `trapizoid.weight`: Numeric vector of trapezoid integration weights at observation angles
 - `kernel.weight`: Denominator from kernel convolution (kernel mass at each observation angle)
 - `fitted.values`: Normalized fitted KDE (activity-scale) at observation angles

- fitted.var: Variance estimate of the fitted values (for SE)
- fitted.se: Standard error of the fitted values
- hour: Corresponding clock time in hours
- coef.cosinor: Named numeric vector with entries:
 - MESOR: the mean of activity density over 24 hours
 - Amplitude: the first-harmonic amplitude derived from Beta and Gamma
 - Acrophase: Acrophase in radians (time-of-peak) of the dominant daily component
 - Acrophase.hr: Acrophase converted to clock hours in [0,24)
 - Beta: coefficient equivalent to the cosine-weighted integral of the KDE
 - Gamma: coefficient equivalent to the sine-weighted integral of the KDE
- post.hoc: Post-hoc peak/trough diagnostics derived from fitted.values at observation angles (MESOR.ph, Bathyphase.ph.time, Trough.ph, Acrophase.ph.time, Peak.ph, Amplitude.ph)

Warnings and edge cases

- The function errors if `all(activity) == 0` or if `activity` contains non-finite values.
- Very small `bw` can cause near-zero denominator values (numerical instability) and produce NA or an error; very large `bw` approaches a near-uniform kernel and will wash out temporal structure.

References

- Cornelissen G. Cosinor-based rhythmometry. Theoretical Biology and Medical Modelling. 2014-12-01 2014;11(1):16. doi:10.1186/1742-4682-11-16
- On Nonparametric Density Estimation for Circular Data: An Overview , bookTitle= Directional Statistics for Innovative Applications: A Bicentennial Tribute to Florence Nightingale. Springer Nature Singapore; 2022:351–378.
- Cremers J, Klugkist I. One Direction? A Tutorial for Circular Data Analysis Using R With Examples in Cognitive Psychology. Methods. Front Psychol. 2018-October-30 2018;9(2040):2040. doi:10.3389/fpsyg.2018.02040

See Also

[CosinorM](#) [ggCosinorM](#)

Examples

```
## Not run:
# Import data
FlyEast

BdfList <-
  BriefSum (
    df = FlyEast,
    SR = 1 / 60,
    Start = "2017-10-24 13:45:00"
  )
```

```

# Let's extract actigraphy data from a single day
df <- BdfList$df
df <- subset (
  x = df,
  subset = df$Date == "2017-10-27"
)

fit <- CosinorM.KDE (
  time = df$Time,
  activity = df$Activity
)

# inspect coefficients
fit$coef.cosinor

# plot KDE in hours
plot (
  x = fit$kdf$hour,
  y = fit$kdf$density,
  type = "l",
  xlab = "Hour",
  ylab = "KDE"
)

## End(Not run)

```

Date2TotalT

Compute Total Time in Each Date (Per Day)

Description

For an ordered vector of Dates or date-times, 'Date2TotalT()' calculates the interval from each element to the next (and for the last element, to the following calendar day at midnight) and returns these durations in the requested time unit.

Usage

```
Date2TotalT(DT, TUnit = "hour", TZ = "local")
```

Arguments

DT	A vector of class Date, POSIXct, or POSIXlt, sorted in ascending order. Each entry represents the start of an interval. Internally, the function appends one extra day beyond the last entry so that the final interval covers a full 24 h until the next midnight. See as.POSIXct.
----	---

TUnit	Character string specifying the unit for the output durations. Must be one of "hour", "minute", or "second". Comparison is case-insensitive. Default is "hour", meaning the returned values are in decimal hours.
TZ	The time zone when the recording started. (default = "local", which means user's local time zone)

Value

A numeric vector of the same length as DT. Each element is the elapsed time between the corresponding entry in DT and the next boundary (either the next date-time in Date, or midnight following the last date), expressed in the units given by TUnit.

Examples

```
# Two calendar days: returns c(24, 24) hours
Date2TotalT (as.Date (c ("2021-01-01", "2021-01-02")), "hour")

# Working in minutes
Date2TotalT (as.POSIXct (c (
  "2021-06-10 08:00:00",
  "2021-06-10 14:30:00"
)), "minute")

# In seconds (case-insensitive unit name)
Date2TotalT (as.Date ("2022-12-31"), "SeCoNd")
```

DateFormat

Detect Possible DT Format

Description

This function will automatically detect possible date format. Users can choose to either reformat the date or report the datetime format detected.

Identifies the likely date format used in character strings and optionally converts these strings to Date objects. This utility is intended to support pre-processing of timestamp data with varied or unknown formatting. When ambiguity arises (e.g., day-month versus month-day ordering), the function defaults to the first compatible format unless explicitly guided by a delimiter or by manual correction.

Usage

```
DateFormat(DT, as.date = TRUE, Delim = NULL)
```

Arguments

DT	A character vector containing date or date-time strings.
as.date	Logical. If TRUE (default), returns converted Date objects. If FALSE, returns the detected format string (e.g., "%d/%m/%Y").
Delim	Optional. A single character (e.g., ".", "-", or "/") to override default date delimiters. Required when input strings may contain nonstandard separators (e.g., "\$", "~").

Value

If as.date = TRUE, returns a vector of class Date.

If as.date = FALSE, returns a character string representing the detected format. When multiple formats coexist or no valid format is detected, warnings are issued and default behavior is applied.

If no format matches, returns NA and issues a warning.

Examples

```
# Consistent format across all strings
DT <- c("2017/05/02", "2000/02/28", "1970/01/02")
DateFormat(DT, as.date = FALSE) # returns parsed Date vector

## Not run:
# Mixed formats within a vector
DT <- c(
  "2017/05/02", "2000.Feb.28", "1970-11-02",
  "January 01, 2025", "December 12, 1980"
)
lapply(DT, DateFormat) # element-wise parsing
### We expect that many of these format will not work because they contain
### text

DT <- c(
  "2017/05/02", "2000.02.28", "1970-11-02",
  "01, 01, 2025", "12, 12, 1980"
)

# Recommended usage for mixed formats:
lapply(DT, DateFormat, Delim = ",") # element-wise parsing

for(x in DT) {
  print(DateFormat(x, Delim = ","))
} # displays format/warning per entry

# Avoid using sapply, because it will convert them into numeric form
sapply(DT, DateFormat)

## End(Not run)
```

DST	<i>Daylight Saving Time</i>
-----	-----------------------------

Description

Determine if Dates Fall in Daylight Saving Time for the Given Time Zone(s)

Usage

```
DST(DT, TZ = "local")
```

Arguments

DT	A vector of class 'Date', 'POSIXct', or 'POSIXlt', representing the dates/times to test. These need not be sorted.
TZ	A single string naming an IANA time zone (e.g. 'America/New_York'). If "local", it will extract local time zone using Sys.timezone . Must match one entry in OlsonNames .

Details

For each element of 'DT', returns 'TRUE' if that date-time is in daylight saving time (DST) under the specified time zone. The function handles transitions by detecting the "gain" or "loss" of hours via 'DST2GL()' and flipping the DST flag on the transition day.

Value

A logical vector of the same length as 'DT'.

- 'TRUE' indicates the timestamp falls in DST
- 'FALSE' indicates standard time.

Examples

```
# Standard use: New York in summer vs. winter
dates <- as.POSIXct (c ("2021-06-15", "2021-12-15"), tz = "America/New_York")
DST (dates, TZ = "America/New_York")

# Around the spring-forward transition
trans <- as.Date (c ("2021-03-13", "2021-03-14", "2021-03-15"))
DST (trans, TZ = "America/New_York")

# Using the local system time zone
DST (as.POSIXct ("2023-07-01 12:00:00"), TZ = "local")
```

DST2GL

*Compute Time Gain/Loss due to Daylight Saving Time***Description**

DST2GL computes, for each date-time entry, how many hours a day deviates from the standard 24hours because of DST shifts. Internally it calls `Date2TotalT()` in seconds, subtracts the 86400s of a normal day, and converts the remainder to hours.

Usage

```
DST2GL(DT, TZ = "local")
```

Arguments

DT	A vector of class <code>Date</code> , <code>POSIXct</code> , or <code>POSIXlt</code> , sorted in ascending order. Each element marks the start of a day boundary. To capture DST transitions, include at least one day before and after the expected shift. Note that the time zone should be specified in the DT. See <code>as.POSIXct</code> .
TZ	The time zone when the recording started. (default = "local", which means user's local time zone)

Value

A numeric vector of the same length as DT. Each value is the number of hours that day is longer (positive) or shorter (negative) than 24h. A zero means no DST shift on that date.

Examples

```
# Example around a typical spring-forward transition (e.g. US second Sunday
# in March)
dates <- as.Date ("2021-03-14")

# On 2021-03-14 clocks jumped forward: day is 23h so output = -1
DST2GL (dates)

# Example around a fall-back transition (e.g. first Sunday in November)
dates <- as.Date ("2021-11-07")

# On 2021-11-07 clocks fall back: day is 25h so output = +1
DST2GL (dates)

# Multiple Dates
sapply (c ("2021-03-13", "2021-03-14", "2021-03-15"), DST2GL)
```

FlyEast

Travel East Sample Data

Description

This sample data is a modified version of a continuous wrist-worn actigraphy recording (Philips ‘ActiWatch5’) collected over a month. The data was originally collected to monitor the influence of jetleg on professional athletes’ performance during competitions.

Usage

```
data(FlyEast)
```

Format

A modified data with 40 thousands plus rows and 3 variables:

Activity The activity count measured by a tri-axis actigraphy

X2 The light exposure data from the light sensor. All values are zero since the function was not disabled during the recording.

Marker A binary indicator generated when the wearer press the button to record onset of sleep or awake.

See Also

[BriefSum TAdjust TravelLog IANA](#)

FlyEast_adj

Adjusted Travel East Sample Data

Description

This is the subset of the travel-adjusted ‘FlyEast’ data. The data was originally collected to monitor the influence of jetleg on professional athletes’ performance during competitions.

Usage

```
data(FlyEast_adj)
```

Format

An adjusted FlyEast data with 40 thousands plus rows and 6 variables:

DateTime The time coordinates of each recorded activities in the standard date and time joint format

Date The date of each recorded activities

Hour The simplified time coordinates in hour form converted from 'DateTime'

Activity The activity count measured by a tri-axis actigraphy

Activity_ID Numeric values denoted the sequence of each activity count in the recording

Note Summary annotation derived from [BriefSum](#).

See Also

[BriefSum](#) [TAdjust](#) [FlyEast](#)

ggActiGlobe

Plot an Overview of an 'ActiGlobe' Activity Time Series

Description

Creates a time-series 'scatterplot' of activity counts from an 'ActiGlobe' data.frame, marking each midnight boundary with a vertical dashed line and 'coloring' points that were flagged (e.g. travel overlaps or unallocated epochs).

Usage

```
ggActiGlobe(df, Bdf, VAct = NULL, VDT = "DateTime")
```

Arguments

df	A data.frame of annotated actigraphy epochs. Must include: - An activity column named by VAct. - A datetime column named by VDT. - Optionally, a 'Note' column to flag affected epochs. See BriefSum and TAdjust for details.
Bdf	A BriefSum object containing per-day metadata for the recording. Note, if jet lag occurred during the recording, please, update the metadata using TAdjust before passing to this function.
VAct	Optional character. Name of the activity column in df. If NULL, defaults to the second column of df.
VDT	Optional character. Name of the POSIXct datetime column in df. If NULL, defaults to "DateTime" of df.

Value

A [ggplot](#) object showing: - Activity counts vs. time. - Dashed vertical lines at each midnight. - Points 'colored' by whether they were flagged in 'Note'.

Examples

```
## Not run:
library(ActiGlobe)

# Overview the Uncorrected Longitudinal Recording
data(FlyEast)

BdfList <-
  BriefSum(
    df = FlyEast,
    SR = 1 / 60,
    Start = "2017-10-24 13:45:00"
  )

p <- ggActiGlobe(
  df = BdfList$df,
  Bdf = BdfList$Bdf,
  VAct = "Activity",
  VDT = "DateTime"
)

print(p)

# Overview the Corrected Longitudinal Recording
data(TLog)

BdfList$Bdf.adj <- TAdjust(BdfList$Bdf, TLog)
p2 <- ggActiGlobe(
  df = BdfList$df,
  Bdf = BdfList$Bdf,
  VAct = "Activity",
  VDT = "DateTime"
)
print(p2)

# Pro-tip: [`cowplot`] can help stack the time series graphs in one
# single plot

## End(Not run)
```

Description

Create visualization of a CosinorM or CosinorM.KDE fit using ggplot2. The plot shows the parametric cosinor fit over a fine time grid, optional pointwise confidence bands, observed data points, MESOR line, acrophase verticals, amplitude annotation segments, and labelled parameter values when requested.

Usage

```
ggCosinorM(
  object,
  labels = TRUE,
  ci = TRUE,
  ci_level = 0.95,
  n = 400,
  point_size = 0.5,
  title_extra = NULL,
  legend.position = "right",
  ...
)
```

Arguments

<code>object</code>	A fitted model of class CosinorM or CosinorM.KDE .
<code>labels</code>	Logical; Default: TRUE, which places repelled labels on the plot with MESOR, amplitude(s), and acrophase(s).
<code>ci</code>	Logical; Default: TRUE, which computes and draws pointwise parametric confidence bands for the fitted cosinor curve using the model covariance.
<code>ci_level</code>	Numeric scalar. The range of the confidence interval, expressed in numeric value between 0 and 1. Default: 0.95
<code>n</code>	Integer scalar. Number of points on the fine prediction grid used to draw the fitted cosinor and confidence ribbon. Default: 400
<code>point_size</code>	Numeric scalar. Plotting size for observed points. Default: 0.5
<code>title_extra</code>	Optional character string appended to the plot title for extra context.
<code>legend.position</code>	Position of the legend on the plot; default is "right". Other options include "top", "bottom", "left", or a numeric vector of length two specifying x and y coordinates.
<code>...</code>	Additional arguments (currently ignored) kept for future update.

Value

A 'ggplot' object representing the cosinor model fit visualization.

See Also

[ggplot](#), [geom_label_repel](#), [predict](#)

Examples

```
## Not run:
# Import data
data (FlyEast)

BdfList <-
```

```

BriefSum (
  df = FlyEast,
  SR = 1 / 60,
  Start = "2017-10-24 13:45:00"
)

# Let's extract actigraphy data from a single day
df <- BdfList$df
df <- subset (df, df$Date == "2017-10-28")

fit <- CosinorM (
  time = df$Time,
  activity = df$Activity,
  tau = 24,
  method = "OLS"
)

p <- ggCosinorM (
  object = fit,
  labels = TRUE,
  ci = TRUE,
  ci_level = 0.95,
  title_extra = "2017-10-24"
)
print (p)

## End(Not run)

```

IANA

IANA Time Table

Description

A dictionary for standard abbreviation and labels of time-zone and their corresponding UTC offset.

Usage

```
data(IANA)
```

Format

A modified IANA 2025b database

Country_Name Full name of the country (e.g., "Canada").

Country_Code Two-letter ISO 3166-1 alpha-2 country code (e.g., "CA").

Timezone_IANA Canonical IANA timezone identifier (e.g., "America/Montreal").

TimeZone_Identifiers Alternative or historical identifiers linked to the IANA timezone.

- TZ_Code** Short code or identifier representing the timezone, often aligned with ISO or custom dataset conventions (e.g., "EST", "PST").
- Offset** Numeric UTC offset in hours, representing the difference from Coordinated Universal Time (e.g., -5, +9).
- Observes_DST** Logical flag indicating whether the timezone observes daylight saving time (TRUE/FALSE).
- Current_DST_Status** Current daylight saving time status at the time of data capture (e.g., "DST Active").
- Current_Abbreviation** Abbreviation in effect at the reference date/time, which may be a daylight saving abbreviation (e.g., "EDT") or a standard abbreviation (e.g., "EST").
- Current_Time_Zone_long_name** Human-readable name of the timezone currently in effect, reflecting whether daylight saving time is active (e.g., "Eastern Daylight Time" in summer, "Eastern Standard Time" in winter).
- Current_Offset** UTC offset in hours at the reference date/time, including any daylight saving adjustment if applicable (e.g., -4 during EDT, -5 during EST).
- Standard_Abbreviation** Abbreviation used during standard (non-daylight saving) time only (e.g., "EST").
- Standard_Time_Zone_long_name** Human-readable name of the timezone during standard time, without daylight saving adjustment (e.g., "Eastern Standard Time").
- Standard_Offset** UTC offset in hours during standard time, excluding daylight saving adjustment (e.g., -5).

Num2UTC	<i>Convert Numbers to Standard UTC Offset</i>
---------	---

Description

Convert numeric values representing UTC offsets (in hours, with optional fractional parts for minutes) into standardized character strings of the form "UTC+HH:MM" or "UTC-HH:MM". This is useful for translating numeric time zone offsets into human-readable UTC offset notation.

Usage

Num2UTC(x)

Arguments

x UTC Offsets <e.g., "UTC+09:30" or "UTC-07:00">

Value

A character vector of standard UTC offsets

See Also

[UTC2Num](#)

Examples

```
# Convert UTC to numeric values
x <- c (9.5, -7)

x1 <- Num2UTC (x)

print (x1)
```

Rad2Hr	<i>Radian to Hour</i>
--------	-----------------------

Description

Convert ‘acrophase’ in radian to clock time based on tau.

Usage

```
Rad2Hr(x, tau)
```

Arguments

x	Numeric vector of phases in radian.
tau	Numeric scalar. Period length in the same time units you want returned (for hours use 24). Must be a single numeric value > 0 and <= 24.

Value

Numeric vector of times in $[0, \tau)$. NA values propagate.

Examples

```
Rad2Hr (pi / 2, tau = 24)

Rad2Hr (c (-pi / 2, 0, pi, 3 * pi / 2), tau = 24)
```

TAdjust

*Adjust Time Shift based on Travel Log***Description**

‘TAdjust()’ is a function that corrects data points and time-shift based on travelling log. Note that it is important to ensure that the UTC-offset value is correct. When in doubt, please use the UTC function or consult IANA table in the package. Daylight saving will not be reassessed as in BriefSum function.

Usage

```
TAdjust(Bdf, TLog, TZ = NULL, fork = FALSE)
```

Arguments

Bdf	A BriefSum object containing per-day metadata for the recording.
TLog	A structured travel log containing date of travel and local time zone. Use ‘TravelLog()’ to generate template.
TZ	The time zone when the recording started. (default = "NULL", which will disregard the use of the initial geographical location-based time zone indicator)
fork	Logical, if TRUE, it will use parallel processing to speed up the computation. Default is FALSE.

Value

A [BriefSum](#) object with adjusted data points and time shift based on travel log.

See Also

[TravelLog](#)

Examples

```
## Not run:

# Import sample data
data (FlyEast)

# Create quick summary of the recording with adjustment for daylight saving.
BdfList <- BriefSum (
  df = FlyEast,
  SR = 1 / 60,
  Start = "2017-10-19 13:45:00",
  TZ = 'America/New_York'
)

# Extract only the summary report
```

```

Bdf <- BdfList$Bdf

# Import sample travel Log
data (TLog)

# Adjust time shift based on travel log
Bdf.adj <- TAdjust (Bdf = Bdf,
                   TLog = TLog,
                   TZ   = 'America/New_York')

# Display the summary
View (Bdf)
View (Bdf.adj) ### Focus on the dates after 2017-11-01

## End(Not run)

```

TimeFormat

Detect Possible Time Format

Description

This function will automatically detect possible time format. Users can choose to either reformat the date or report the datetime format detected.

Usage

```
TimeFormat(time, as.time = FALSE)
```

Arguments

<code>time</code>	A character string of time.
<code>as.time</code>	A binary operator indicating whether to return a converted time based on the detection or the time format. (default: FALSE, which returns time format)

Value

If `as.time = TRUE`, returns a character vector of the input times reformatted according to the detected format. Each element corresponds to the respective entry in `time`.

If `as.time = FALSE`, returns a character scalar giving the best-matching time format string.

If no format matches, returns NA and issues a warning.

Examples

```

# Example 1: When all dates have the same format
## Create and store a date in a variable called Time
Time <- c (
  "2017/05/02 23:00:01", "1970/01/02 05:10:33",

```

```

      "2000/02/28 07:00:00"
    )

    ## Ask TimeFormat to tell us the format of the time.
    TimeFormat (time = Time, as.time = FALSE)

    # Example 2: When multiple formats co-exist in a variable
    ## Create and store dates and time in a variable called Time
    Time <- c (
      "2017/05/02 23:00:01", "2000/02/28 07:00", "1970/01/02",
      "2022/11/28 08:35 PM"
    )

    ## Ask TimeFormat to tell us the format of the Time.
    ### DO This!
    ##### Option 1.
    print (lapply (Time, TimeFormat))

    ##### Option 2. To match the warning message to the items
    for (x in Time) {
      print (TimeFormat (x))
    }

    ### DO NOT!
    ### print(TimeFormat(time = Time))
    ### Note, this process will fail because there are multiple formats

```

TLog

Travel Log

Description

A modified travel log documenting the dates when the wearer departs for long-distance travel.

Usage

```
data(TLog)
```

Format

A data frame with over 40,000 rows and 5 variables:

ID Pseudonym of the wearer.

UTC_Offset UTC offset of the wearer's location or destination.

Country_with_Daylight_Saving Binary indicator (TRUE/FALSE) for whether daylight saving time is observed at the location or destination.

date_Start Start date of the wearer's initial location or date of departure.

date_End Optional date when the wearer departs again from the initial location or previous destination.

See Also

[TAdjust TravelLog IANA](#)

TravelLog	<i>Generate the Travel Log Template Needed for Time Zone Correction</i>
-----------	---

Description

Generate the Travel Log Template Needed for Time Zone Correction

Usage

```
TravelLog(Write = FALSE, Dir = NULL)
```

Arguments

- | | |
|-------|--|
| Write | A binary code to indicate whether to write a .csv file containing the template needed for the travel log. (default = FALSE) When set to FALSE, the template will be returned as an object. When set to TRUE, user must provide the target directory where they wish to store the template. |
| Dir | The directory where the travel log template to be exported <e.g. "C:/Users/____YOUR USERNAME____/UPSTREAM FOLDER/.../FOLDER NAME/"> |

Value

a travel-log template as a data.frame or written as a CSV file

Examples

```
Tlg <- TravelLog (Write = FALSE)

print (Tlg)
```

TZ2UTC	<i>Compute UTC offset based on Time Zone and Date</i>
--------	---

Description

Compute UTC offset based on Time Zone and Date

Usage

```
TZ2UTC(DT, TZ = "local")
```

Arguments

DT	The date in the format as "2021-03-05"
TZ	The time zone when the recording started. Default = "local", which means user's local time zone

Value

A character string indicating the UTC offset, e.g., "UTC-05:00"

Examples

```
x <- as.Date (c ("2017-10-24", "2017-11-20"))
TZ2UTC (DT = x, TZ = "America/New_York") ## Expect two different UTCs

## Not run:
# A vector of dates
x <-
  seq.Date (
    from = as.Date ("2017-10-24"),
    to = as.Date ("2017-11-27"),
    by = "day"
  )

# If the user resides in a country that follows daylight saving time
TZ2UTC (DT = x, TZ = "local")

# If not
# TZ2UTC(DT = x, TZ = "America/New_York")

## End(Not run)
```

UTC2Num	<i>Convert Standard UTC Offset to Numbers</i>
---------	---

Description

Convert Standard UTC Offset to Numbers

Usage

```
UTC2Num(x)
```

Arguments

x	Character string of UTC offsets, e.g., "UTC+09:30" or "UTC-07:00".
---	--

Value

Numeric values of UTC offsets

Examples

```
# Convert UTC to numeric values
x <- c("UTC+09:30", "UTC-07:00")

x1 <- UTC2Num(x)

print(x)
```

 UTCwDST

Determine if Daylight Saving Transitions may occur in an UTC Offset

Description

Tests whether any IANA time zone associated with a given UTC offset undergoes daylight saving time (DST) transitions. The function identifies all zones matching the specified UTC offset and evaluates whether a time change occurs between winter and summer timestamps.

Usage

```
UTCwDST(UTCs, DT = NULL, fork = FALSE)
```

Arguments

UTCs	A character vector or numeric vector representing UTC offsets. Accepted formats include "UTC+08:00", "UTC-05:00", or numeric values like "+8", "-5", etc. The function internally maps UTC strings to numeric offsets using UTC2Num .
DT	A POSIXct date/time used as the reference point for offset comparison. Defaults to January 1, 2021 UTC if 'NULL'.
fork	Logical, if TRUE, it will use parallel processing to speed up the computation. Default is FALSE.

Details

For each value in UTCs, the function retrieves the corresponding IANA time zones whose offset matches the specified UTC string or numeric hour. It then compares the DST status of each zone on January 1 and July 15. If at least one zone shows a shift in DST status between the two dates, the function returns 'TRUE' for that offset.

Value

A logical vector the same length as UTCs. Each entry is 'TRUE' if at least one time zone at the specified offset undergoes a DST transition, 'FALSE' otherwise.

See Also

[DST UTC2Num OlsonNames](#)

Examples

```
## Not run:
# Check for DST transitions in UTC+1 and UTC+8
UTCwDST (UTCs = c ("UTC+01:00", "UTC+08:00"))

# Use numeric offset directly
UTCwDST (UTCs = c (1, 8))

# UTC-5 commonly includes DST zones (e.g., New York)
UTCwDST (UTCs = -5)

## End(Not run)
```

write.act

Export Daily Activity Score and Recording Summary

Description

A function that exports two types of files locally. 1. A summary of the entire recording (.csv). 2. Daily activity recordings in individual files (.txt). The default delimiter is set to be space, with ‘.’ as the decimal indicator. As such, it should be noted that some transformation may be needed when opening in EXCEL on a computer with non-English EU settings.

Usage

```
write.act(
  Dir,
  ID,
  df,
  Bdf,
  TUnit = "hour",
  VAct = NULL,
  VTm = NULL,
  Incomplete = FALSE,
  Travel = TRUE,
  Simple = FALSE
)
```

Arguments

Dir	The directory where the recordings to be exported <e.g. "C:/Users/___YOUR USERNAME___/UPSTREAM FOLDER/.../FOLDER NAME/">
ID	The subject's ID which would be used to create a folder.

df	A data.frame of raw actigraphy recording. Both time and activity count should be included in the 'df'. See 'VAct' and 'VTm' for further detail.
Bdf	A BriefSum object. Note, if jet lag occurred during the recording, please, update the metadata using TAdjust before passing to this function.
TUnit	Character; time-unit for the x-axis of each day's timeline. Must be one of 'day', 'hour', 'minute' or 'second'. Default is 'hour'.
VAct	Optional character. Name of the activity column in 'df'. If NULL, defaults to the second column of 'df'.
VTm	Optional character. Name of the time index column in 'df'. If NULL, defaults to the first column of 'df'.
Incomplete	Logical; if TRUE, days flagged 'Incomplete Recording' (i.e. <24 h) are retained in the data list with recordings segmented by day. Default = FALSE (these days are removed).
Travel	Logical; if TRUE, days flagged 'Travel' are retained although some data points from an earlier adjacent calendar day may be duplicated (a warning is issued). If FALSE, travel days and the day before/after are excluded. Default = TRUE.
Simple	Logical; if TRUE, only columns stored in the original recordings will be exported. If FALSE, all information stored in 'df' will be generated. Default = FALSE

Value

Text files ".txt" containing segmented recordings of each day and 'Bdf' from [BriefSum](#)

- Text File: Recordings segmented by the recording date
- Summary: Bdf exported as ".csv" file.

Examples

```
## Not run:

# Import data
data (FlyEast)

# Create quick summary of the recording with adjustment for daylight saving.
BdfList <-
  BriefSum (
    df = FlyEast,
    SR = 1 / 60,
    Start = "2017-10-24 13:45:00"
  )

# Let's extract the quick summary of the recording
Bdf <- BdfList$Bdf

### Note that since the original data was affected by travel-induced time
### shift, the recordings would not be properly segmented from 2017-11-02.
```

```
## To avoid time shift due to travelling, we will keep only the first 8 days.
Bdf <- Bdf [1:8, ]

df <- BdfList$df
# Export daily recordings and the summary report

write.act (
  Dir = getwd (), ## Export to the current working directory
  ID = "JD",
  df = df,
  Bdf = Bdf,
  VAct = "Activity",
  VTm = "Time"
)

## To see what was exported copy the path generated by the code below and
## paste it as an url in a web browser or the file explorer.
getwd ()

## End(Not run)
```

write.cosinor

Generate Report for Daily Cosinor and Generate Summary Plot

Description

A multi-wrapper function which generate data trend using [geom_smooth](#) and model plot using [CosinorM](#) or [CosinorM.KDE](#). All daily results could be found in the exported report. A summary file in .csv format is also generated to update the original 'Bdf' with cosinor model coefficients for each day.

Usage

```
write.cosinor(
  Dir,
  ID,
  DailyAct,
  Bdf,
  VAct = NULL,
  VTm = NULL,
  method = "OLS",
  tau = 24,
  ph = FALSE,
  overwrite = FALSE
)
```

Arguments

Dir	The directory where the recordings to be exported <e.g. "C:/Users/___YOUR USERNAME___/UPSTREAM FOLDER/.../FOLDER NAME/">
ID	The subject's ID which would be used to create a folder.
DailyAct	A data.frame list generated by 'Act2Daily()'
Bdf	A BriefSum object. Note, if jet lag occurred during the recording, please, update the metadata using TAdjust before passing to this function.
VAct	Optional character. Name of the activity column in each data.frame object ('df') in 'DailyAct'. If NULL, defaults to the second column of 'df'.
VTm	Optional character. Name of the activity column in each data.frame object ('df') in 'DailyAct'. If NULL, defaults to the first column of 'df'.
method	Character string specifying estimation method • "OLS": Ordinary least square cosinor model via CosinorM. (default) • "FGLS": Feasible generalized least squares cosinor model via CosinorM. • "KDE": Gaussian kernel density estimate (KDE) on circular time (hours of day) via CosinorM.KDE.
tau	Numeric scalar or vector for the assumed circadian period, passed to CosinorM . Default is 24 for single-phase; multiple phases can be supplied via [c()].
ph	Optional logical scaler; if only a single period is assigned to tau, TRUE will force the function to append the post-hoc circadian estimates instead of the model estimates. (default: FALSE)
overwrite	Logical; if TRUE, existing files with the same name will be overwritten. Default = FALSE.

Value

Creates a multi-page '.pdf' visual document and a text-based '.csv' summary.

- PDF A multi-page PDF written to 'pdfDir' (one file per 'ID'). Each page corresponds to a recording day and contains two panels: the top panel shows the raw activity (points) and a smoothed trend; the bottom panel shows the model fit for the same day
- 'Summary.csv' An updated 'Bdf' documentation with one row per recording day. Model-based circadian parameters are appended at the end of the file for each day.

See Also

[Act2Daily](#) [CosinorM](#) [CosinorM.KDE](#)

Examples

```
## Not run:

# Import data
data (FlyEast)
```

```

# Create quick summary of the recording with adjustment for daylight saving.
BdfList <-
  BriefSum (
    df = FlyEast,
    SR = 1 / 60,
    Start = "2017-10-24 13:45:00"
  )

# Let's extract the quick summary of the recording
Bdf <- BdfList$Bdf
df <- BdfList$df

### Note that since the original data was affected by travel-induced time
### shift, the recordings would not be properly segmented from 2017-11-02.

## To avoid time shift due to travelling, we will keep only the first 8 days.
Bdf <- Bdf [1:8, ]

## Segment Data by Day
dfList <-
  Act2Daily (
    df = df,
    Bdf = Bdf,
    VAct = "Activity",
    VTm = "Time",
    Incomplete = TRUE,
    Travel = TRUE
  )

## Generate graphic summary of daily recordings in pdf with an addition
## `Summary` file just like write.act
write.cosinor (
  Dir = getwd (), ## Export to the current working directory
  ID = "JD",
  DailyAct = dfList$Daily_df,
  Bdf = Bdf,
  VAct = "Activity",
  VTm = "Time"
)

## To see what was exported copy the path generated by the code below and
## paste it as an url in a web browser or the file explorer.
getwd ()

## End(Not run)

```

Index

- * **Actigraphy**
 - Act2Daily, [3](#)
 - TAdjust, [28](#)
 - write.act, [34](#)
- * **Adjust**
 - TAdjust, [28](#)
- * **Daily**
 - Act2Daily, [3](#)
- * **Export**
 - write.act, [34](#)
- * **FlyEast_adj**
 - FlyEast_adj, [21](#)
- * **FlyEast**
 - FlyEast, [21](#)
- * **IANA**
 - IANA, [25](#)
- * **KDE**
 - CosinorM.KDE, [12](#)
- * **Log**
 - TravelLog, [31](#)
- * **Segment**
 - Act2Daily, [3](#)
- * **Shift**
 - TravelLog, [31](#)
- * **TZ**
 - TZ2UTC, [31](#)
- * **Template**
 - TravelLog, [31](#)
- * **Timezone**
 - TravelLog, [31](#)
- * **Travel**
 - TravelLog, [31](#)
- * **UTC**
 - TZ2UTC, [31](#)
 - UTC2Num, [32](#)
- * **acrophase**
 - Rad2Hr, [27](#)
- * **actigraphy**
 - BriefSum, [7](#)
 - ggActiGlobe, [22](#)
 - write.cosinor, [36](#)
- * **adjust**
 - FlyEast_adj, [21](#)
- * **bootstrap**
 - boot.seci, [5](#)
- * **boot**
 - boot.seci, [5](#)
- * **circadian**
 - CosinorM.KDE, [12](#)
- * **circular**
 - CosinorM.KDE, [12](#)
- * **ci**
 - boot.seci, [5](#)
- * **cosinor**
 - CosinorM, [8](#)
 - CosinorM.KDE, [12](#)
 - ggCosinorM, [23](#)
 - write.cosinor, [36](#)
- * **daylight**
 - DST, [19](#)
 - TLog, [30](#)
- * **diary**
 - TLog, [30](#)
- * **dst**
 - DST, [19](#)
- * **export**
 - write.cosinor, [36](#)
- * **ggplot**
 - ggCosinorM, [23](#)
- * **graph**
 - ggCosinorM, [23](#)
- * **num**
 - UTC2Num, [32](#)
- * **offset**
 - UTC2Num, [32](#)
- * **plot**
 - ggCosinorM, [23](#)
- * **radian**

Rad2Hr, [27](#)
 * **saving**
 TLog, [30](#)
 * **se**
 boot.seci, [5](#)
 * **summary**
 BriefSum, [7](#)
 * **timezone**
 TLog, [30](#)
 * **travel**
 TLog, [30](#)
 * **trend**
 write.cosinor, [36](#)
 * **visualization**
 ggActiGlobe, [22](#)
 * **write**
 write.cosinor, [36](#)

 Act2Daily, [3](#), [37](#)
 as.POSIXct, [7](#)

 boot, [6](#)
 boot.seci, [5](#)
 BriefSum, [3](#), [7](#), [21](#), [22](#), [28](#), [35](#), [37](#)

 CosinorM, [8](#), [15](#), [24](#), [36](#), [37](#)
 CosinorM.KDE, [11](#), [12](#), [24](#), [36](#), [37](#)

 Date2TotalT, [16](#)
 DateFormat, [7](#), [17](#)
 DST, [19](#), [34](#)
 DST2GL, [7](#), [20](#)

 FlyEast, [21](#), [22](#)
 FlyEast_adj, [21](#)

 geom_label_repel, [24](#)
 geom_smooth, [36](#)
 ggActiGlobe, [22](#)
 ggCosinorM, [11](#), [15](#), [23](#)
 ggplot, [22](#), [24](#)

 IANA, [21](#), [25](#), [31](#)

 lm, [9–11](#)

 Num2UTC, [26](#)

 OlsonNames, [19](#), [34](#)

 predict, [24](#)

 Rad2Hr, [27](#)

 Sys.timezone, [19](#)

 TAdjust, [3](#), [4](#), [21](#), [22](#), [28](#), [31](#), [35](#), [37](#)
 TimeFormat, [7](#), [29](#)
 TLog, [30](#)
 TravelLog, [21](#), [28](#), [31](#), [31](#)
 TZ2UTC, [31](#)

 UTC2Num, [26](#), [32](#), [33](#), [34](#)
 UTCwDST, [33](#)

 vcovHC, [9](#)

 write.act, [4](#), [34](#)
 write.cosinor, [36](#)