Package: accelerometry (via r-universe)

September 5, 2024

Type Package

Title Functions for Processing Accelerometer Data

Version 3.1.3

License GPL-3

LazyData true

Date 2018-12-01

Author Dane R. Van Domelen

Maintainer Dane R. Van Domelen <vandomed@gmail.com>

Description A collection of functions that perform operations on time-series accelerometer data, such as identify non-wear time, flag minutes that are part of an activity bout, and find the maximum 10-minute average count value. The functions are generally very flexible, allowing for a variety of algorithms to be implemented. Most of the functions are written in C++ for efficiency.

Depends R (>= 3.0.0)

Imports Rcpp (>= 0.12.15), dvmisc, extraDistr

Suggests knitr, rmarkdown, pander

LinkingTo Rcpp

RoxygenNote 6.0.1

VignetteBuilder knitr

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

RemoteUrl https://github.com/vandomed/accelerometry

RemoteRef HEAD

RemoteSha 70afcac78b9dfc3c061a2e19d93cd66de8d6b8d9

Contents

`accelerometry` *Functions for Processing Accelerometer Data*

Description

A collection of functions that perform operations on time-series accelerometer data, such as identify non-wear time, flag minutes that are part of an activity bout, and find the maximum 10-minute average count value. The functions are generally very flexible, allowing for a variety of algorithms to be implemented. Most of the functions are written in C++ for efficiency.

Details

Package:	accelerometry
Type:	Package
Version:	3.1.3
Date:	2018-12-01
License:	GPL-3

See CRAN documentation for full list of functions.

Author(s)

Dane R. Van Domelen
`<vandomed@gmail.com>`

References

Centers for Disease Control and Prevention (CDC). National Center for Health Statistics (NCHS). National Health and Nutrition Examination Survey Data. Hyattsville, MD: US Department of Health and Human Services, Centers for Disease Control and Prevention, 2003-6. Available at: https://wwwn.cdc.gov/nchs/nhanes/Default.aspx. Accessed Aug. 19, 2018.

Eddelbuettel, D. and Francois, R. (2011) Rcpp: Seamless R and C++ Integration. Journal of Statistical Software, 40(8), 1-18. http://www.jstatsoft.org/v40/i08/.

Eddelbuettel, D. (2013) Seamless R and C++ Integration with Rcpp. Springer, New York. ISBN 978-1-4614-6867-7.

Eddelbuettel, D. and Balamuta, J.J. (2017). Extending R with C++: A Brief Introduction to Rcpp. PeerJ Preprints 5:e3188v1. https://doi.org/10.7287/peerj.preprints.3188v1.

National Cancer Institute. Risk factor monitoring and methods: SAS programs for analyzing NHANES 2003-2004 accelerometer data. Available at: http://riskfactor.cancer.gov/tools/nhanes_pam. Accessed Aug. 19, 2018.

Van Domelen, D.R., Pittard, W.S. and Harris, T.B. (2018) nhanesaccel: Process accelerometer data from NHANES 2003-2006. R package version 3.1.1. https://github.com/vandomed/accelerometry.

Acknowledgment: This material is based upon work supported by the National Science Foundation Graduate Research Fellowship under Grant No. DGE-0940903.

`adherence_dodd` *Estimate a Participant's Probability of Adhering to "N Days Per Week" Type of Physical Activity Guideline (Dodd's Method)*

Description

Implements the Bayesian approach developed by Dodd and used in the landmark Troiano et al. paper (*MSSE* 2008).

Usage

```
adherence_dodd(n, x, n.rec = 7, x.rec = 5, posterior = NULL)
```

Arguments

`n`	Number of monitoring days.
`x`	Number of active days.
`n.rec`	Denominator for recommendation.
`x.rec`	Numerator for recommendation.
`posterior`	Can be `NULL` for original Dodd method or `"mean"` or `"median"` for modified version described above.

Details

The approach aims to estimate a participant's probability of meeting guidelines of the form "at least x minutes per day for at least y days per week" based on observing X active days out of n monitoring days. We illustrate here with the "5+ active days per week" guideline that motivated the approach.

The prior assumption for the participant's *daily* adherence probability is:

p_d ~ Uni(0, 1)

Given p_d, the number of active days out of n monitoring days is distributed:

X|p_d ~ Bin(n, p_d)

It can be shown that the posterior for p_d is:

p_d|X ~ Beta(X + 1, n - X + 1)

Under a somewhat questionable independence assumption, the weekly adherence probability is p_w = P(Y >= 5) with Y ~ Bin(7, p_d). Dodd estimates p_w as:

p_w.hat = P(p_d >= 5/7 | X)

which can be calculated using `pbeta`.

In my view, the quantity P(p_d >= 5/7 | X) is not a good estimator for p_w. Consider what would happen in a really long protocol. The Beta posterior for p_d would be very tightly centered around the true p_d, and p_w.hat = P(p_d >= 5/7 | X) would be very close to either 0 or 1 – not very close to what we're trying to estimate, p_w.

A solution is to define p_d.hat as the posterior mean, median, or mode, and map that estimate to p_w, i.e. p_w.hat = P(Y >= 5) with Y ~ Bin(7, p_d.hat). So there is an option for that.

References

Dodd, K. (2008). Estimation of the population prevalence of adherence to physical activity recommendations based on NHANES accelerometry measurements. Technical Report. Available at: https://epi.grants.cancer.gov/nhanes_pam/bayesian_adherence_estimation.pdf. Accessed Nov. 13, 2018.

Troiano, R.P., Berrigan, D., Dodd, K.W., Masse, L.C. and McDowell, M. (2008). Physical activity in the United States measured by accelerometer. Medicine \& Science in Sports \& Exercise 40(1): 181–188.

Examples

```
# Generate data from hypothetical study with 1000 subjects, valid days
# randomly sampled from 1-7, and p_d's drawn from Beta(0.5, 3).
set.seed(1)
n <- sample(1: 7, size = 1000, replace = TRUE)
p_d <- rbeta(n = 1000, shape1 = 0.5, shape2 = 3)
x <- rbinom(n = 1000, size = n, prob = p_d)

# Estimate p_w's using Dodd's method
p_w.hat <- adherence_dodd(n = n, x = x)

# Note that the mean p_w.hat differs considerably from the true mean p_w,
# reflecting bias in the estimator.
mean(p_w.hat)
```

```
mean(pbinom(q = 4, size = 7, prob = p_d, lower.tail = FALSE))
```

`adherence_garriguet` *Estimate a Participant's Probability of Adhering to "N Days Per Week" Type of Physical Activity Guideline (Garriguet's Method)*

Description

Implements the Bayesian approach described by Garriguet (*Statistics Canada* 2016).

Usage

```
adherence_garriguet(n, x, alpha, beta, n.rec = 7, x.rec = 5)
```

Arguments

`n`	Number of monitoring days.
`x`	Number of exercise days.
`alpha`	Parameter in p_d ~ Beta(alpha, beta). Corresponds to `shape1` in `Beta` functions.
`beta`	Parameter in p_d ~ Beta(alpha, beta). Corresponds to `shape2` in `Beta` functions.
`n.rec`	Denominator for recommendation.
`x.rec`	Numerator for recommendation.

Details

The approach aims to estimate a participant's probability of meeting guidelines of the form "at least x minutes per day for at least y days per week" based on observing X active days out of n monitoring days.

The prior assumption for the participant's *daily* adherence probability is:

p_d ~ Beta(alpha, beta)

where alpha and beta are estimated via maximum likelihood using the observed sample proportions if active days for all study participants. This can be done separately via `mles_beta`.

Given p_d, the number of active days out of n monitoring days is distributed:

X|p_d ~ Bin(n, p_d)

It can be shown that the posterior for p_d is:

p_d|X ~ Beta(alpha2 = alpha + X, beta2 = beta + n - X)

Garriguet then uses the Beta-binomial distribution, which describes binomial data with success probability randomly drawn from Beta(alpha, beta). The weekly adherence estimator is defined as:

p_w.hat <- P(Y >= 5) with Y ~ Betabin(7, alpha2, beta2)

which can be calculated using `mles_beta`.

References

Garriguet, D. (2016). Using a betabinomial distribution to estimate the prevalence of adherence to physical activity guidelines among children and youth. Statistics Canada, Catalogue no. 82-003-X. Health Reports 27(4): 3-9. Available at: https://www150.statcan.gc.ca/n1/pub/82-003-x/2016004/article/14489-eng.pdf.

Examples

```
# Generate data from hypothetical study with 1000 subjects, valid days
# randomly sampled from 1-7, and p_d's drawn from Beta(0.5, 3).
set.seed(1)
n <- sample(1: 7, size = 1000, replace = TRUE)
p_d <- rbeta(n = 1000, shape1 = 0.5, shape2 = 3)
x <- rbinom(n = 1000, size = n, prob = p_d)

# First step: Estimate (alpha, beta) via maximum likelihood. Have to change
# 0's to 0.01 and 1's to 0.99 to avoid Inf's
p_d.hat <- x / n
p_d.hat[p_d.hat == 0] <- 0.01
p_d.hat[p_d.hat == 1] <- 0.99
mles <- mles_beta(x = p_d.hat)

# Estimate each subject's weekly adherence probability
p_w.hat <- adherence_garriguet(n = n, x = x, alpha = mles$par[1], beta = mles$par[2])

# Note that the mean p_w.hat differs considerably from the true mean p_w,
# reflecting bias in the estimator.
mean(p_w.hat)
mean(pbinom(q = 4, size = 7, prob = p_d, lower.tail = FALSE))
```

`artifacts` *Accelerometer Artifact Correction*

Description

Corrects abnormally high count values in accelerometer data by replacing such values with the average of neighboring count values. Returns integer vector despite the average calculation often producing a decimal; this follows the convention used in the NCI's SAS programs (http://riskfactor.cancer.gov/tools/nhanes_pam).

Usage

```
artifacts(counts, thresh, counts_classify = NULL)
```

Arguments

- `counts` Integer vector with accelerometer count values.
- `thresh` Integer value specifying the smallest count value that should be considered an artifact.
- `counts_classify` Integer vector with accelerometer count values to base artifact classification on, but not to adjust. Mainly included for triaxial data, where you might want to define artifacts based on vertical-axis counts but then actually adjust the triaxial sum or vector magnitude counts.

Value

Integer vector equivalent to `counts` except where artifacts were adjusted.

References

National Cancer Institute. Risk factor monitoring and methods: SAS programs for analyzing NHANES 2003-2004 accelerometer data. Available at: http://riskfactor.cancer.gov/tools/nhanes_pam. Accessed Aug. 19, 2018.

Examples

```
# Load accelerometer data for first 5 participants in NHANES 2003-2004
data(unidata)

# Get data from ID number 21007
counts.part3 <- unidata[unidata[, "seqn"] == 21007, "paxinten"]

# Replace counts > 10,000 with average of neighboring values
counts.part3.corrected <- artifacts(counts = counts.part3, thresh = 10000)
```

`blockaves` *Block Averages*

Description

Calculates block averages (i.e. moving averages but for non-overlapping intervals) or maximum block average. For optimal speed, use `integer = TRUE` if `x` is an integer vector and `integer = FALSE` otherwise. If `length(x)` is not an exact multiple of `window`, the last partial segment is dropped.

Usage

```
blockaves(x, window, integer = FALSE, max = FALSE)
```

Arguments

`x`	Integer or numeric vector.
`window`	Integer value specifying window length.
`integer`	Logical value for whether `x` is an integer vector.
`max`	Logical value for whether to return maximum moving average (as opposed to vector of moving averages).

Value

Numeric value or vector depending on `max`.

Examples

```
# Load accelerometer data for first 5 participants in NHANES 2003-2004
data(unidata)

# Get data from ID number 21005, Saturday only
counts.sat <- unidata[unidata[, "seqn"] == 21005 & unidata[, "paxday"] == 7,
                      "paxinten"]

# Calculate and plot hourly count averages
hourly.averages <- blockaves(x = counts.sat, window = 60, integer = TRUE)
plot(hourly.averages)
```

`blocksums` *Block Sums*

Description

Calculates block sums (i.e. moving sums but for non-overlapping intervals) or maximum block sum. For optimal speed, use `integer = TRUE` if `x` is an integer vector and `integer = FALSE` otherwise. If `length(x)` is not an exact multiple of `window`, the last partial segment is dropped.

Usage

```
blocksums(x, window, integer = FALSE, max = FALSE)
```

Arguments

`x`	Integer or numeric vector.
`window`	Integer value specifying window length.
`integer`	Logical value for whether `x` is an integer vector.
`max`	Logical value for whether to return maximum moving average (as opposed to vector of moving averages).

Value

Numeric value or vector depending on max.

Examples

```
# Load accelerometer data for first 5 participants in NHANES 2003-2004
data(unidata)

# Get data from ID number 21005, Saturday only
counts.sat <- unidata[unidata[, "seqn"] == 21005 & unidata[, "paxday"] == 7,
                      "paxinten"]

# Calculate and plot hourly count sums
hourly.sums <- blocksums(x = counts.sat, window = 60, integer = TRUE)
plot(hourly.sums)
```

---

bouts — *Physical Activity Bout Detection*

Description

Identify bouts of physical activity based on a vector of accelerometer count values.

Usage

```
bouts(counts, weartime = NULL, bout_length = 10L, thresh_lower = 0L,
  thresh_upper = 100000L, tol = 0L, tol_lower = 0L, tol_upper = 100000L,
  nci = FALSE, days_distinct = FALSE)
```

Arguments

counts	Integer vector with accelerometer count values.
weartime	Integer vector with 1's for wear time minutes and 0's for non-wear time minutes.
bout_length	Integer value specifying minimum length of an activity bout.
thresh_lower	Integer value specifying lower bound for count values to be included for the intensity level.
thresh_upper	Integer value specifying upper bound for count values to be included for the intensity level.
tol	Integer value specifying number of minutes with count values outside of [thresh_lower, thresh_upper] to allow during an activity bout.
tol_lower	Integer value specifying lower cut-off for count values outside of intensity range during an activity bout.
tol_upper	Integer value specifying upper cut-off for count values outside of intensity range during an activity bout.

nci	Logical value for whether to use algorithm from NCI's SAS programs. See **Details**.
days_distinct	Logical value for whether to treat each day of data as distinct, i.e. identify non-wear time and activity bouts for day 1, then day 2, etc. If FALSE, algorithm is applied to full monitoring period continuously. If protocol has participants remove accelerometer for sleep, strongly recommend setting to FALSE to capture non-wear periods that start between 11 pm and midnight. Function assumes that first 1440 data points are day 1, next 1440 are day 2, and so on.

Details

If nci = FALSE, the algorithm uses a moving window to go through every possible interval of length bout_length in counts. Any interval in which all counts are >= tol_lower and <= tol_upper, and no more than tol counts are less than thresh_lower or greater than thresh_upper, is classified as an activity bout.

If nci = TRUE, activity bouts are classified according to the algorithm used in the NCI's SAS programs. Briefly, this algorithm defines an activity bout as an interval of length bout_length that starts with a count value in [thresh_lower, thresh_upper] and has no more than tol counts outside of that range. If these criteria are met, the bout continues until there are (tol + 1) consecutive minutes outside of [thresh_lower, thresh_upper]. The parameters tol_lower and tol_upper are not used.

If the user allows for a tolerance (e.g. tol = 2) and does not use the NCI algorithm (i.e. nci = FALSE), specifying a non-zero value for tol_lower is highly recommended. Otherwise the algorithm will tend to classify minutes immediately before and after an activity bout as being part of the bout.

Specifying thresh_lower while using an arbitrarily large value for thresh_upper is generally recommended. Specifying both of these parameters can be overly restrictive in that the algorithm may miss bouts of activity in which counts are consistently high, but not exclusively in one intensity range.

Value

Integer vector with 1's for minutes that are part of an activity bout and 0's for minutes that are not.

References

National Cancer Institute. Risk factor monitoring and methods: SAS programs for analyzing NHANES 2003-2004 accelerometer data. Available at: http://riskfactor.cancer.gov/tools/nhanes_pam. Accessed Aug. 19, 2018.

Acknowledgment: This material is based upon work supported by the National Science Foundation Graduate Research Fellowship under Grant No. DGE-0940903.

Examples

```
# Load accelerometer data for first 5 participants in NHANES 2003-2004
data(unidata)

# Get data from ID number 21005
```

```
counts.part1 <- unidata[unidata[, "seqn"] == 21005, "paxinten"]

# Identify periods of valid wear time
wear.part1 <- weartime(counts = counts.part1)

# Identify moderate-to-vigorous activity bouts
mvpa.bouts <- bouts(counts = counts.part1, weartime = wear.part1,
                    thresh_lower = 2020)
```

`cut_counts` *Cut Count Values into Intensity Ranges*

Description

Given a vector of accelerometer count values, classifies each count value into intensity level 1, 2, 3, 4, or 5 (typically representing sedentary, light, lifestyle, moderate, and vigorous).

Usage

```
cut_counts(counts, int_cuts = as.integer(c(100, 760, 2020, 5999)))
```

Arguments

`counts`	Integer vector with accelerometer count values.
`int_cuts`	Numeric vector with four cutpoints from which five intensity ranges are derived. For example, `int_cuts = c(100, 760, 2020, 5999)` creates: 0-99 = intensity 1; 100-759 = intensity level 2; 760-2019 = intensity 3; 2020-5998 = intensity 4; >= 5999 = intensity 5.

Value

Integer vector.

Examples

```
# Load accelerometer data for first 5 participants in NHANES 2003-2004
data(unidata)

# Get data from ID number 21005
counts.part1 <- unidata[unidata[, "seqn"] == 21005, "paxinten"]

# Cut into 5 intensity levels and plot
intensity.part1 <- cut_counts(counts = counts.part1)
plot(intensity.part1)
```

intensities *Physical Activity Intensities*

Description

Given a vector of accelerometer count values, calculates time spent in 5 mutually exclusive user-defined intensity levels (typically representing sedentary, light, lifestyle, moderate, and vigorous) as well as the total counts accumulated in various intensities. Non-wear time should be removed from `counts` before calling `intensities` to avoid overestimating sedentary time.

Usage

```
intensities(counts, int_cuts = as.integer(c(100, 760, 2020, 5999)))
```

Arguments

`counts`	Integer vector with accelerometer count values.
`int_cuts`	Numeric vector with four cutpoints from which five intensity ranges are derived. For example, `int_cuts = c(100, 760, 2020, 5999)` creates: 0-99 = intensity 1; 100-759 = intensity level 2; 760-2019 = intensity 3; 2020-5998 = intensity 4; >= 5999 = intensity 5.

Value

Integer vector of length 16 in which the first eight values are minutes in intensities 1, 2, 3, 4, 5, 2-3, 4-5, and 2-5, and the next eight are counts accumulated during time spent in each of those intensities.

Examples

```
# Load accelerometer data for first 5 participants in NHANES 2003-2004
data(unidata)

# Get data from ID number 21005
counts.part1 <- unidata[unidata[, "seqn"] == 21005, "paxinten"]

# Create vector of counts during valid wear time only
counts.part1.wear <- counts.part1[weartime(counts = counts.part1) == 1]

# Calculate physical activity intensity variables
intensity.variables <- intensities(counts = counts.part1.wear)
```

`inverse_rle2` *Inverse Run Length Encoding (Alternate Implementation)*

Description

Re-constructs vector compressed by `rle2`.

Usage

```
inverse_rle2(x)
```

Arguments

`x`	Object returned by `rle2`.

Value

Integer or numeric vector.

Examples

```
# Create dummie vector x
x <- c(0, 0, 0, -1, -1, 10, 10, 4, 6, 6)

# Summarize x using rle2
x.summarized <- rle2(x)

# Reconstruct x
x.reconstructed <- inverse_rle2(x.summarized)
```

`mles_beta` *Get Maximum Likelihood Estimates for Beta Distribution*

Description

Same idea as `fitdistr` function in **MASS**, but has default starting values and uses `nlminb` rather than `optim`.

Usage

```
mles_beta(x, start = c(0.5, 0.5))
```

Arguments

`x`	Observations assumed to be iid Beta(alpha, beta).
`start`	Starting values for alpha and beta.

Examples

```
# Generate data from Beta(1, 2) and get MLE's
set.seed(1)
x <- rbeta(n = 1000, shape1 = 1, shape2 = 2)
mles <- mles_beta(x)
mles$par
```

`movingaves` *Moving Averages*

Description

Calculates moving averages or maximum moving average. For optimal speed, use `integer = TRUE` if `x` is an integer vector and `integer = FALSE` otherwise.

Usage

```
movingaves(x, window, integer = FALSE, max = FALSE)
```

Arguments

`x`	Integer or numeric vector.
`window`	Integer value specifying window length.
`integer`	Logical value for whether `x` is an integer vector.
`max`	Logical value for whether to return maximum moving average (as opposed to vector of moving averages).

Value

Numeric value or vector depending on `max`.

Examples

```
# Load accelerometer data for first 5 participants in NHANES 2003-2004
data(unidata)

# Get data from ID number 21005
id.part1 <- unidata[unidata[, "seqn"] == 21005, "seqn"]
counts.part1 <- unidata[unidata[, "seqn"] == 21005, "paxinten"]

# Create vector of all 10-minute moving averages
all.movingaves <- movingaves(x = counts.part1, window = 10, integer = TRUE)

# Calculate maximum 10-minute moving average
max.movingave <- movingaves(x = counts.part1, window = 10, integer = TRUE,
                            max = TRUE)
```

personvars — *Calculating Daily Averages for Physical Activity Variables*

Description

Not intended for direct use.

Usage

```
personvars(dayvars, rows, days, wk, we)
```

Arguments

`dayvars`	Numeric matrix with daily physical activity variables.
`rows`	Numeric value specifying number of rows in the matrix to be returned.
`days`	Integer value specifying minimum number of valid days a participant must have to be included.
`wk`	Integer value specifying minimum number of valid weekdays a participant must have to be included.
`we`	Integer value specifying minimum number of valid weekend days a participant must have to be included.

Value

Numeric matrix.

process_tri — *Process Triaxial Minute-to-Minute Accelerometer Data*

Description

Calculates a variety of physical activity variables based on triaxial minute-to-minute accelerometer count values for individual participants. Assumes first 1440 minutes are day 1, next 1440 are day 2, and so on. If final day has less than 1440 minutes, it is excluded. A data dictionary for the variables created is available here: `https://github.com/vandomed/accelerometry/blob/master/process_tri_dictionary.csv`.

Usage

```
process_tri(counts, steps = NULL, nci_methods = FALSE, start_day = 1,
  start_date = NULL, id = NULL, brevity = 1, hourly_var = "cpm",
  hourly_wearmin = 0, hourly_normalize = FALSE, valid_days = 1,
  valid_wk_days = 0, valid_we_days = 0, int_axis = "vert",
  int_cuts = c(100, 760, 2020, 5999), cpm_nci = FALSE,
  days_distinct = FALSE, nonwear_axis = "vert", nonwear_window = 60,
  nonwear_tol = 0, nonwear_tol_upper = 99, nonwear_nci = FALSE,
  weartime_minimum = 600, weartime_maximum = 1440,
  active_bout_length = 10, active_bout_tol = 0, mvpa_bout_tol_lower = 0,
  vig_bout_tol_lower = 0, active_bout_nci = FALSE, sed_bout_tol = 0,
  sed_bout_tol_maximum = int_cuts[2] - 1, artifact_axis = "vert",
  artifact_thresh = 25000, artifact_action = 1, weekday_weekend = FALSE,
  return_form = "daily")
```

Arguments

`counts` Integer matrix with three columns of count values, e.g. vertical-axis counts, anteroposterior (AP)-axis counts, and mediolateral (ML)-axis counts.

`steps` Integer vector with steps.

`nci_methods` Logical value for whether to set all arguments so as to replicate the data processing methods used in the NCI's SAS programs. More specifically:

valid_days = 4

valid_wk_days = 0

valid_we_days = 0

int_axis = "vert"

int_cuts = c(100, 760, 2020, 5999)

cpm_nci = TRUE

days_distinct = TRUE

nonwear_axis = "vert"

nonwear_window = 60

nonwear_tol = 2

nonwear_tolupper = 100

nonwear_nci = TRUE

weartime_minimum = 600

weartime_maximum = 1440

active_bout_length = 10

active_bout_tol = 2

mvpa_bout_tol_lower = 0

vig_bout_tol_lower = 0

active_bout_nci = TRUE

sed_bout_tol = 0

sed_bout_tol_maximum = 759

artifact_thresh = 32767

artifact_action = 3
If TRUE, you can still specify non-default values for brevity and weekday_weekend.

start_day
: Integer value specifying day of week for first day of monitoring, with 1 = Sunday, ..., 7 = Satuday.

start_date
: Date for first day of monitoring, which function can use to figure out start_day.

id
: Numeric value specifying ID number of participant.

brevity
: Integer value controlling the number of physical activity variables generated. Choices are 1 for basic indicators of physical activity volume, 2 for addditional indicators of activity intensities, activity bouts, sedentary behavior, and peak activity, and 3 for additional hourly count averages.

hourly_var
: Character string specifying what hourly activity variable to record, if brevity = 3. Choices are "counts_vert", "counts_ap", "counts_ml", "counts_sum", "counts_vm", "cpm_vert", "cpm_ap", "cpm_ml", "sed_min", "sed_bouted_10min", and "sed_breaks".

hourly_wearmin
: Integer value specifying minimum number of wear time minutes needed during a given hour to record a value for the hourly activity variable.

hourly_normalize
: Logical value for whether to normalize hourly activity by number of wear time minutes.

valid_days
: Integer value specifying minimum number of valid days to be considered valid for analysis.

valid_wk_days
: Integer value specifying minimum number of valid weekdays to be considered valid for analysis.

valid_we_days
: Integer value specifying minimum number of valid weekend days to be considered valid for analysis.

int_axis
: Character string specifying which axis should be used to classify intensities. Choices are "vert", "ap", "ml", "sum" (for triaxial sum), and "vm (for triaxial vector magnitude).

int_cuts
: Numeric vector with four cutpoints from which five intensity ranges are derived. For example, int_cuts = c(100, 760, 2020, 5999) creates: 0-99 = intensity 1; 100-759 = intensity level 2; 760-2019 = intensity 3; 2020-5998 = intensity 4; >= 5999 = intensity 5. Intensities 1-5 are typically viewed as sedentary, light, lifestyle, moderate, and vigorous.

cpm_nci
: Logical value for whether to calculate average counts per minute by dividing average daily counts by average daily wear time, as opposed to taking the average of each day's counts per minute value. Strongly recommend leave as FALSE unless you wish to replicate the NCI's SAS programs.

days_distinct
: Logical value for whether to treat each day of data as distinct, as opposed to analyzing the entire monitoring period as one continuous segment.

nonwear_axis
: Character string specifying which axis should be used to classify non-wear time. Choices are "vert", "ap", "ml", "sum" (for triaxial sum), and "vm" (for triaxial vector magnitude).

nonwear_window
: Integer value specifying minimum length of a non-wear period.

nonwear_tol
: Integer value specifying tolerance for non-wear algorithm, i.e. number of minutes with non-zero counts allowed during a non-wear interval.

nonwear_tol_upper
: Integer value specifying maximum count value for a minute with non-zero counts during a non-wear interval.

nonwear_nci
: Logical value for whether to use non-wear algorithm from NCI's SAS programs.

weartime_minimum
: Integer value specifying minimum number of wear time minutes for a day to be considered valid.

weartime_maximum
: Integer value specifying maximum number of wear time minutes for a day to be considered valid. The default is 1440, but you may want to use a lower value (e.g. 1200) if participants were instructed to remove devices for sleeping, but often did not.

active_bout_length
: Integer value specifying minimum length of an active bout.

active_bout_tol
: Integer value specifying number of minutes with counts outside the required range to allow during an active bout. If non-zero and `active_bout_nci = FALSE`, specifying non-zero values for `mvpa_bout_tol_lower` and `vig_bout_tol_lower` is highly recommended. Otherwise minutes immediately before and after an active bout will tend to be classified as part of the bout.

mvpa_bout_tol_lower
: Integer value specifying lower cut-off for count values outside of required intensity range for an MVPA bout.

vig_bout_tol_lower
: Integer value specifying lower cut-off for count values outside of required intensity range for a vigorous bout.

active_bout_nci
: Logical value for whether to use algorithm from the NCI's SAS programs for classifying active bouts.

sed_bout_tol
: Integer value specifying number of minutes with counts outside sedentary range to allow during a sedentary bout.

sed_bout_tol_maximum
: Integer value specifying upper cut-off for count values outside sedentary range during a sedentary bout.

artifact_axis
: Character string specifying which axis should be used to identify artifacts (impossibly high count values). Choices are "vert", "ap", "ml", "sum" (for triaxial sum), and "vm" (for triaxial vector magnitude).

artifact_thresh
: Integer value specifying the smallest count value that should be considered an artifact.

artifact_action
: Integer value controlling method of correcting artifacts. Choices are 1 to exclude days with one or more artifacts, 2 to lump artifacts into non-wear time, 3 to replace artifacts with the average of neighboring count values, and 4 to take no action.

weekday_weekend
: Logical value for whether to calculate averages for weekdays and weekend days separately (in addition to all valid days).

return_form
: Character string controlling how variables are returned. Choices are "daily" for per-day summaries, "averages" for averages across all valid days, and "both" for a list containing both.

Value

Numeric matrix or list of two numeric matrices, depending on `return_form`.

References

National Cancer Institute. Risk factor monitoring and methods: SAS programs for analyzing NHANES 2003-2004 accelerometer data. Available at: http://riskfactor.cancer.gov/tools/nhanes_pam. Accessed Aug. 19, 2018.

Examples

```
# Note that the 'tridata' dataset contains 7 days of fake triaxial
# accelerometer data

# Process data using default parameters and request per-day variables
accel.days <- process_tri(
  counts = tridata,
  return_form = "daily"
)

# Repeat, but request averages across all valid days
accel.averages <- process_tri(
  counts = tridata,
  return_form = "averages"
)

# Create per-day summary again, but with many more variables
accel.days2 <- process_tri(
  counts = tridata,
  brevity = 2,
  return_form = "daily"
)
names(accel.days2)
```

`process_uni` *Process Uniaxial Minute-to-Minute Accelerometer Data*

Description

Calculates a variety of physical activity variables based on uniaxial minute-to-minute accelerometer count values for individual participants. Assumes first 1440 minutes are day 1, next 1440 are day 2, and so on. If final day has less than 1440 minutes, it is excluded. A data dictionary for the variables created is available here: https://github.com/vandomed/accelerometry/blob/master/process_uni_dictionary.csv.

Usage

```
process_uni(counts, steps = NULL, nci_methods = FALSE, start_day = 1,
  start_date = NULL, id = NULL, brevity = 1, hourly_var = "cpm",
  hourly_wearmin = 0, hourly_normalize = FALSE, valid_days = 1,
  valid_wk_days = 0, valid_we_days = 0, int_cuts = c(100, 760, 2020,
  5999), cpm_nci = FALSE, days_distinct = FALSE, nonwear_window = 60,
  nonwear_tol = 0, nonwear_tol_upper = 99, nonwear_nci = FALSE,
  weartime_minimum = 600, weartime_maximum = 1440,
  active_bout_length = 10, active_bout_tol = 0, mvpa_bout_tol_lower = 0,
  vig_bout_tol_lower = 0, active_bout_nci = FALSE, sed_bout_tol = 0,
  sed_bout_tol_maximum = int_cuts[2] - 1, artifact_thresh = 25000,
  artifact_action = 1, weekday_weekend = FALSE, return_form = "averages")
```

Arguments

`counts`	Integer vector with accelerometer count values.
`steps`	Integer vector with steps.
`nci_methods`	Logical value for whether to set all arguments so as to replicate the data processing methods used in the NCI's SAS programs. More specifically: valid_days = 4 valid_wk_days = 0 valid_we_days = 0 int_cuts = c(100, 760, 2020, 5999) cpm_nci = TRUE days_distinct = TRUE nonwear_window = 60 nonwear_tol = 2 nonwear_tolupper = 100 nonwear_nci = TRUE weartime_minimum = 600 weartime_maximum = 1440 active_bout_length = 10 active_bout_tol = 2 mvpa_bout_tol_lower = 0 vig_bout_tol_lower = 0 active_bout_nci = TRUE sed_bout_tol = 0

sed_bout_tol_maximum = 759
artifact_thresh = 32767
artifact_action = 3
If TRUE, you can still specify non-default values for brevity and weekday_weekend.

start_day
: Integer value specifying day of week for first day of monitoring, with 1 = Sunday, ..., 7 = Satuday.

start_date
: Date for first day of monitoring, which function can use to figure out start_day.

id
: Numeric value specifying ID number of participant.

brevity
: Integer value controlling the number of physical activity variables generated. Choices are 1 for basic indicators of physical activity volume, 2 for addditional indicators of activity intensities, activity bouts, sedentary behavior, and peak activity, and 3 for additional hourly count averages.

hourly_var
: Character string specifying what hourly activity variable to record, if brevity = 3. Choices are "counts", "cpm", "sed_min", "sed_bouted_10min", and "sed_breaks".

hourly_wearmin
: Integer value specifying minimum number of wear time minutes needed during a given hour to record a value for the hourly activity variable.

hourly_normalize
: Logical value for whether to normalize hourly activity by number of wear time minutes.

valid_days
: Integer value specifying minimum number of valid days to be considered valid for analysis.

valid_wk_days
: Integer value specifying minimum number of valid weekdays to be considered valid for analysis.

valid_we_days
: Integer value specifying minimum number of valid weekend days to be considered valid for analysis.

int_cuts
: Numeric vector with four cutpoints from which five intensity ranges are derived. For example, int_cuts = c(100, 760, 2020, 5999) creates: 0-99 = intensity 1; 100-759 = intensity level 2; 760-2019 = intensity 3; 2020-5998 = intensity 4; >= 5999 = intensity 5. Intensities 1-5 are typically viewed as sedentary, light, lifestyle, moderate, and vigorous.

cpm_nci
: Logical value for whether to calculate average counts per minute by dividing average daily counts by average daily wear time, as opposed to taking the average of each day's counts per minute value. Strongly recommend leave as FALSE unless you wish to replicate the NCI's SAS programs.

days_distinct
: Logical value for whether to treat each day of data as distinct, as opposed to analyzing the entire monitoring period as one continuous segment.

nonwear_window
: Integer value specifying minimum length of a non-wear period.

nonwear_tol
: Integer value specifying tolerance for non-wear algorithm, i.e. number of minutes with non-zero counts allowed during a non-wear interval.

nonwear_tol_upper
: Integer value specifying maximum count value for a minute with non-zero counts during a non-wear interval.

nonwear_nci
: Logical value for whether to use non-wear algorithm from NCI's SAS programs.

weartime_minimum
: Integer value specifying minimum number of wear time minutes for a day to be considered valid.

weartime_maximum
: Integer value specifying maximum number of wear time minutes for a day to be considered valid. The default is 1440, but you may want to use a lower value (e.g. 1200) if participants were instructed to remove devices for sleeping, but often did not.

active_bout_length
: Integer value specifying minimum length of an active bout.

active_bout_tol
: Integer value specifying number of minutes with counts outside the required range to allow during an active bout. If non-zero and active_bout_nci = FALSE, specifying non-zero values for mvpa_bout_tol_lower and vig_bout_tol_lower is highly recommended. Otherwise minutes immediately before and after an active bout will tend to be classified as part of the bout.

mvpa_bout_tol_lower
: Integer value specifying lower cut-off for count values outside of required intensity range for an MVPA bout.

vig_bout_tol_lower
: Integer value specifying lower cut-off for count values outside of required intensity range for a vigorous bout.

active_bout_nci
: Logical value for whether to use algorithm from the NCI's SAS programs for classifying active bouts.

sed_bout_tol
: Integer value specifying number of minutes with counts outside sedentary range to allow during a sedentary bout.

sed_bout_tol_maximum
: Integer value specifying upper cut-off for count values outside sedentary range during a sedentary bout.

artifact_thresh
: Integer value specifying the smallest count value that should be considered an artifact.

artifact_action
: Integer value controlling method of correcting artifacts. Choices are 1 to exclude days with one or more artifacts, 2 to lump artifacts into non-wear time, 3 to replace artifacts with the average of neighboring count values, and 4 to take no action.

weekday_weekend
: Logical value for whether to calculate averages for weekdays and weekend days separately (in addition to all valid days).

return_form
: Character string controlling how variables are returned. Choices are "daily" for per-day summaries, "averages" for averages across all valid days, and "both" for a list containing both.

Value

Numeric matrix or list of two numeric matrices, depending on return_form.

References

National Cancer Institute. Risk factor monitoring and methods: SAS programs for analyzing NHANES 2003-2004 accelerometer data. Available at: http://riskfactor.cancer.gov/tools/nhanes_pam. Accessed Aug. 19, 2018.

Examples

```
# Note that the 'unidata' dataset contains accelerometer data for first 5
# subjects in NHANES 2003-2004

# Get data from ID number 21005
id.part1 <- unidata[unidata[, "seqn"] == 21005, "seqn"]
counts.part1 <- unidata[unidata[, "seqn"] == 21005, "paxinten"]

# Process data from ID 21005 and request per-day variables
accel.days <- process_uni(
  counts = counts.part1,
  id = id.part1,
  return_form = "daily"
)

# Repeat, but request averages across all valid days
accel.averages <- process_uni(
  counts = counts.part1,
  id = id.part1,
  return_form = "averages"
)

# Process data according to methods used in NCI's SAS programs
accel.nci1 <- process_uni(
  counts = counts.part1,
  id = id.part1,
  brevity = 2,
  valid_days = 4,
  cpm_nci = TRUE,
  days_distinct = TRUE,
  nonwear_tol = 2,
  nonwear_tol_upper = 100,
  nonwear_nci = TRUE,
  weartime_maximum = 1440,
  active_bout_tol = 2,
  active_bout_nci = TRUE,
  artifact_thresh = 32767,
  artifact_action = 3,
  return_form = "averages"
)

# Repeat, but use nci_methods input for convenience
accel.nci2 <- process_uni(
  counts = counts.part1,
  id = id.part1,
  nci_methods = TRUE,
```

```
  brevity = 2,
  return_form = "averages"
)

# Results are identical
all.equal(accel.nci1, accel.nci2)
```

---

rle2 *Run Length Encoding (Alternate Implementation)*

Description

Summarizes vector containing runs of repeated values. Very similar to rle, but sometimes much faster, and with an option to return the start/end indices for each run.

Usage

```
rle2(x, class = NULL, indices = FALSE)
```

Arguments

x	Vector (see class).
class	Character string specifying class of x. If unspecified, function figures it out (at cost of slightly slower run time).
indices	Logical value for whether to record start/stop positions in addition to values and lengths for each run.

Value

Integer or numeric matrix.

Examples

```
# Create dummie vector x
x <- c(0, 0, 0, -1, -1, 10, 10, 4, 6, 6)

# Summarize x using rle2
x.summarized <- rle2(x)

# Repeat, but also record start/stop indices for each run
x.summarized <- rle2(x = x, indices = TRUE)
```

sedbreaks *Sedentary Breaks*

Description

Identifies sedentary breaks in accelerometer count data.

Usage

```
sedbreaks(counts, weartime = NULL, thresh = 100, flags = FALSE)
```

Arguments

`counts`	Integer vector with accelerometer count values.
`weartime`	Integer vector with 1's for wear time minutes and 0's for non-wear time minutes.
`thresh`	Integer value specifying minimum count value to consider a break from sedentary time.
`flags`	Logical value for whether to return a vector of 1's and 0's flagging the sedentary breaks (as opposed to the total number of sedentary breaks).

Value

Integer value or vector depending on `flags`.

Examples

```
# Load accelerometer data for first 5 participants in NHANES 2003-2004
data(unidata)

# Get data from ID number 21005
id.part1 <- unidata[unidata[, "seqn"] == 21005, "seqn"]
counts.part1 <- unidata[unidata[, "seqn"] == 21005, "paxinten"]

# Identify periods of valid wear time
wear.part1 <- weartime(counts = counts.part1)

# Count number of sedentary breaks (over full week)
n.sedbreaks <- sedbreaks(counts = counts.part1, weartime = wear.part1)

# Flag sedentary breaks
sedbreaks.flagged <- sedbreaks(counts = counts.part1, weartime = wear.part1,
                               flags = TRUE)
```

tridata — *Triaxial Sample Data*

Description

Toy dataset with triaxial minute-to-minute counts generated from a trivariate normal distribution. Does not closely resemble real accelerometer data.

unidata — *Uniaxial Sample Data*

Description

Accelerometer data for the first 5 participants in the National Health and Nutrition Examination Survey (NHANES) 2003-2004 dataset.

Source

https://wwwn.cdc.gov/nchs/nhanes/search/datapage.aspx?Component=Examination&CycleBeginYear=2003

weartime — *Wear Time Classification*

Description

Classifies wear time vs. non-wear time based on a vector of accelerometer count values.

Usage

```
weartime(counts, window = 60L, tol = 0L, tol_upper = 99L, nci = FALSE,
  days_distinct = FALSE, units_day = 1440L)
```

Arguments

counts	Integer vector with accelerometer count values.
window	Integer value specifying minimum length of a non-wear period.
tol	Integer value specifying tolerance for non-wear algorithm, i.e. number of seconds/minutes with non-zero counts allowed during a non-wear interval.
tol_upper	Integer value specifying maximum count value for a second/minute with non-zero counts during a non-wear interval.

`nci` Logical value for whether to use algorithm from NCI's SAS programs. See **Details**.

`days_distinct` Logical value for whether to treat each day of data as distinct, as opposed to analyzing the entire monitoring period as one continuous segment. For minute-to-minute counts, strongly recommend setting to `FALSE` to correctly classify time near midnight.

`units_day` Integer value specifying how many data point are in a day. Typically either 1440 or 86400 depending on whether count values are minute-to-minute or second-to-second.

Details

If `nci = FALSE`, the algorithm uses a moving window to go through every possible interval of length `window` in `counts`. Any interval in which no more than `tol` counts are non-zero, and those are still < `tol.upper`, is classified as non-wear time.

If `nci = TRUE`, non-wear time is classified according to the algorithm used in the NCI's SAS programs. Briefly, this algorithm defines a non-wear period as an interval of length `window` that starts with a count value of 0, does not contain any periods with `(tol + 1)` consecutive non-zero count values, and does not contain any counts > `tol.upper`. If these criteria are met, the non-wear period continues until there are `(tol + 1)` consecutive non-zero count values or a single count value > `tol.upper`.

Value

Integer vector with 1's for valid wear time and 0's for non-wear time.

References

National Cancer Institute. Risk factor monitoring and methods: SAS programs for analyzing NHANES 2003-2004 accelerometer data. Available at: http://riskfactor.cancer.gov/tools/nhanes_pam. Accessed Aug. 19, 2018.

Acknowledgment: This material is based upon work supported by the National Science Foundation Graduate Research Fellowship under Grant No. DGE-0940903.

Examples

```
# Load accelerometer data for first 5 participants in NHANES 2003-2004
data(unidata)

# Get data from ID number 21005
counts.part1 <- unidata[unidata[, "seqn"] == 21005, "paxinten"]

# Identify periods of valid wear time
weartime.flag <- weartime(counts = counts.part1)
```

Index