Package ‘PhysicalActivity’
January 22, 2021

Version 0.2-4
Date 2020-12-29
Title Process Accelerometer Data for Physical Activity Measurement
Description It provides a function `wearingMarking` for classification of monitor wear and nonwear time intervals in accelerometer data collected to assess physical activity. The package also contains functions for making plot for accelerometer data and obtaining the summary of various information including daily monitor wear time and the mean monitor wear time during valid days. `deliveryPred` and `markDelivery` can classify days for ActiGraph delivery by mail; `deliveryPreprocess` can process accelerometry data for analysis by zeropadding incomplete days and removing low activity days; `markPAI` can categorize physical activity intensity level based on user-defined cut-points of accelerometer counts. It also supports importing ActiGraph AGD files with `readActigraph` and `queryActigraph` functions.
License GPL (>= 3)
Depends R (>= 2.10)
Suggests RSQLite, DBI, data.table, e1071, keras, randomForest, reticulate, rms
RoxygenNote 7.1.1
NeedsCompilation no
Author Leena Choi [aut, cre] (<https://orcid.org/0000-0002-2544-7090>), Cole Beck [aut] (<https://orcid.org/0000-0002-6849-6255>), Zhouwen Liu [aut], Ryan Moore [aut], Charles E. Matthews [aut], Maciej S. Buchowski [aut]
Maintainer Leena Choi <leena.choi@Vanderbilt.Edu>
Repository CRAN
Date/Publication 2021-01-22 22:30:02 UTC

R topics documented:
PhysicalActivity-package ................................................... 2
PhysicalActivity-package

Process Accelerometer Data for Physical Activity Measurement

Description

It provides a function `wearingMarking` for classification of monitor wear and nonwear time intervals in accelerometer data collected to assess physical activity. The package also contains functions for making plot for accelerometer data and obtaining the summary of various information including daily monitor wear time and the mean monitor wear time during valid days.

Details

The revised package version 0.2-2 improved the functions in the previous version regarding speed and robustness. In addition, several functions were added: `markDelivery` can classify days for ActiGraph delivery by mail; `markPAI` can categorize physical activity intensity level based on user-defined cut-points of accelerometer counts. It also supports importing ActiGraph AGD files with `readActigraph` and `queryActigraph` functions. The package also better supports time zones and daylight saving.

Classify wear and nonwear time status for accelerometer data by epoch-by-epoch basis by `wearingMarking`.

Classify mail delivery and non-delivery day status for accelerometer data by `markDelivery`.

Three options are available for the package: `pa.validCut=600`, `pa.timeStamp='TimeStamp'`, and `pa.cts='axis'`. When these options are specified (as in `markDelivery`), the other functions will automatically respect these values as defaults. For instance, the count variable in `data(dataSec)` is "counts". Running options(`pa.cts='counts'`) allows the user to avoid specifying the "cts" argument in `wearingMarking`. The options for validCut and timeStamp are rarely changed.

Shiny app called `Actigraph` can be used to visualize accelerometer data and summarize the data. Please see [https://github.com/couthcommander/PhysicalActivityShiny](https://github.com/couthcommander/PhysicalActivityShiny).
dataCollapser

Author(s)

Leena Choi <leena.choi@Vanderbilt.Edu>, Cole Beck <cole.beck@vumc.org>, Zhouwen Liu <zhouwen.liu@vumc.org>, Charles E. Matthews <Charles(Matthews2@nih.gov>, and Maciej S. Buchowski <maciej.buchowski@Vanderbilt.Edu>

Maintainer: Leena Choi <leena.choi@Vanderbilt.Edu>

References


Examples

data(dataSec)

mydata1m = dataCollapser(dataSec, TS = "TimeStamp", col = "counts", by = 60)
options(pa.cts = 'counts') # change cnt variable from "axis1" to "counts"

plotData(data=mydata1m, frame = 90)

summaryData(data=mydata1m, validCut=600, perMinuteCts=1, markingString = "w")

---

dataCollapser  Collapse Accelerometer Data to a Dataset with a Longer Epoch

Description

The function collapses counts in data collected with a short epoch to make a data set with a longer epoch. For example, this function collapses data with 1-sec epoch to 10-sec epoch or 1-min epoch data.

Usage

dataCollapser(dataset, TS, by, col, func = sum, ...)

### Arguments

- **dataset**: The source dataset, in dataframe format, that needs to be collapsed.
- **TS**: The column name for timestamp.
- **by**: Epoch in seconds for a collapsed dataset. For example, to collapse second data to minute data, set by = 60; to collapse 10-second data to minute data, set by = 60.
- **col**: The column name(s) to collapse. If not provided, will default to all numeric columns.
- **func**: A method for collapsing counts. The default is the summation of counts.
- **...**: Argument settings that to be used by user-defined "func" setting.

### Value

A collapsed data with user specified epoch.

### Author(s)

Zhouwen Liu <zhouwen.liu@vumc.org>

### References


### Examples

```r
# collapse 1-sec epoch data to 10-sec epoch data
mydata10s = dataCollapser(dataSec, TS = "TimeStamp", col = "counts", by = 10)

# collapse 1-sec epoch data to 1-min epoch data
mydata1m = dataCollapser(dataSec, TS = "TimeStamp", col = "counts", by = 60)
```

### Accelerometer Data Example

Approximately 3 days of accelerometer data collected with 1-sec epoch in the correct data format that can be used by `wearingMarking` to classify wear and nonwear time.

### Usage

```r
data(dataSec)
```
**deliveryData**

**Format**

A data frame with 238140 observations on the following 2 required variables.

- **TimeStamp** A character vector, timestamp of accelerometer measurements
- **counts** A numeric vector, counts as accelerometer measurements

**References**


**Examples**

```r
data(dataSec)
```

---

**deliveryData**  
*Data Example for Mail Delivery Day Classification*

**Description**

Approximately 15 days of 3-axis accelerometer data collected with 1-minute epoch.

**Usage**

```r
data(deliveryData)
```

**Format**

A data frame with 20987 observations on the following variables.

- **TimeStamp** A character vector, timestamp of accelerometer measurements
- **axis1** A numeric vector, counts from axis1 of 3-axis accelerometer
- **axis2** A numeric vector, counts from axis2 of 3-axis accelerometer
- **axis3** A numeric vector, counts from axis3 of 3-axis accelerometer
- **steps** A numeric vector, the number of steps
- **vm** A numeric vector, the vector magnitude calculated from counts of axis1, axis2 and axis3

**Author(s)**

Cole Beck <cole.beck@vumc.org>, Leena Choi <leena.choi@Vanderbilt.Edu>

**Examples**

```r
data(deliveryData)
```
**deliveryFeatures**

**Description**

The function extracts multiple statistical features relevant for classification of days as delivery or human wear. The extracted features are: mean, variance, maximum, absolute change, absolute energy, proportion of trial completed, 95th quantile, skewness, and kurtosis.

**Usage**

deliveryFeatures(df, ...)

**Arguments**

- **df**
  A dataframe. The source accelerometry dataset, in dataframe format.
- **...**
  not used at this time

**Details**

Function works for data consisting of one or multiple unique trials.

**Value**

A dataframe is returned with a row for each unique day and a column for each feature.

**Note**

The input dataframe should have the following columns: ‘TimeStamp’, ‘axis1’, ‘axis2’, ‘axis3’, ‘vm’, where ‘vm’ is the vector magnitude of axes 1, 2, and 3. Dataframe should also be formatted to 60 second epoch.

**Author(s)**

Ryan Moore <ryan.moore@vumc.org>, Cole Beck <cole.beck@vumc.org>, and Leena Choi <leena.choi@Vanderbilt.Edu>

**See Also**

deliveryPred

**Examples**

data(deliveryData)

deliveryDataProcessed <- deliveryPreprocess(df = deliveryData)
deliveryDataFeats <- deliveryFeatures(df = deliveryDataProcessed)
Wrapper Function for Accelerometry data Preprocessing, Feature Extraction, and Delivery Prediction

Description

The function is a wrapper function that performs preprocessing, feature extraction, and delivery day prediction of an accelerometry dataset. The prediction model can be selected from one of three models, a Random Forest, a logistic regression, and a convolutional neural network (default: Random Forest).

Usage

`deliveryPred(df, model = c("RF", "NN", "GLM"))`

Arguments

- `df` A dataframe. The source accelerometry dataset, in dataframe format.
- `model` A character. Indicates which prediction model to use. ‘RF’ is a Random Forest. ‘GLM’ is a logistic regression, and ‘NN’ is a convolutional neural network.

Details

Function works for data consisting of one or multiple unique trials.

Value

A dataframe is returned with a predicted probability of each day being a delivery activity day.

Note

The input dataframe should have the following columns: ‘TimeStamp’, ‘axis1’, ‘axis2’, ‘axis3’, ‘vm’, where ‘vm’ is the vector magnitude of axes 1, 2, and 3. Dataframe should also be formatted to 60 second epoch.

The function uses the default preprocessing criteria used in the development of the predictive models.

Author(s)

Ryan Moore <ryan.moore@vumc.org>, Cole Beck <cole.beck@vumc.org>, and Leena Choi <leena.choi@Vanderbilt.Edu>

See Also

deliveryPreprocess, deliveryFeatures, deliveryPrediction
Examples

data(deliveryData)

predictions <- deliveryPred(df = deliveryData, model = "GLM")

Description

The function predicts the probability of each day in an accelerometry dataset being caused from
delivery activity instead of human activity. The prediction model can be selected from one of
three models, a Random Forest, a logistic regression, and a convolutional neural network (default:
Random Forest).

Usage

deliveryPrediction(df, feats, model = c("RF", "GLM", "NN"), ...)

Arguments

df A dataframe. The source accelerometry dataset, in dataframe format.
feats A dataframe. Features output from the deliveryFeatures function.
model A character. Indicates which prediction model to use. ‘RF’ is a Random Forest.
‘GLM’ is a logistic regression, and ‘NN’ is a convolutional neural network.
... not used at this time

Details

Function works for data consisting of one or multiple unique trials.

Value

A dataframe is returned with a predicted probability of each day being a delivery activity day.

Note

The input dataframe should have the following columns: ‘TimeStamp’, ‘axis1’, ‘axis2’, ‘axis3’,
‘vm’, where ‘vm’ is the vector magnitude of axes 1, 2, and 3. Dataframe should also be formatted
to 60 second epoch.

Author(s)

Ryan Moore <ryan.moore@vumc.org>, Cole Beck <cole.beck@vumc.org>, and Leena Choi <leena.choi@Vanderbilt.Edu>
See Also
deliveryFeatures, deliveryPred

Examples
data(deliveryData)
deliveryDataProcessed <- deliveryPreprocess(df = deliveryData)
deliveryDataFeats <- deliveryFeatures(df = deliveryDataProcessed)
deliveryPrediction(deliveryDataProcessed, deliveryDataFeats)

Description
This function preprocesses accelerometry data by removing days based on a total activity count (default: less than 5000) or total time with activity (default: less than 10 minutes). Additionally, the function has an option to zeropad truncated days such that days that do not have a whole day of 1440 minutes of data spanning from 00:00 to 23:59 (default: TRUE).

Usage
deliveryPreprocess(df, minLow = 5000, minTime = 10, zeropad = TRUE, ...)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>df</td>
<td>A dataframe. The source accelerometry dataset, in dataframe format.</td>
</tr>
<tr>
<td>minLow</td>
<td>Numeric. The minimum total counts of movement for a day to not be removed.</td>
</tr>
<tr>
<td>minTime</td>
<td>Numeric. The minimum number of minutes of activity for a day to not be removed.</td>
</tr>
<tr>
<td>zeropad</td>
<td>Boolean value for truncated days to be zeropadded.</td>
</tr>
<tr>
<td>...</td>
<td>Not used at this time</td>
</tr>
</tbody>
</table>

Details
Function works for dataframes from one or multiple unique trials.

Value
The dataframe is returned with days fulfilling the dropping criteria removed and truncated days zeropadded. A new column indicating which day is added to the dataframe.
markDelivery

Note
The input dataframe should have the following columns: ‘TimeStamp’, ‘axis1’, ‘axis2’, ‘axis3’, ‘vm’, where ‘vm’ is the vector magnitude of axes 1, 2, and 3. Dataframe should also be formatted to 60 second epoch.

Author(s)
Ryan Moore <ryan.moore@vumc.org>, Cole Beck <cole.beck@vumc.org>, and Leena Choi <leena.choi@Vanderbilt.Edu>

See Also
deliveryPred

Examples

data(deliveryData)
deliveryDataProcessed <- deliveryPreprocess(df = deliveryData)

Usage
markDelivery(
data,
cts = getOption("pa.cts"),
markingString = "w",
window = c("trim", "consecutive", "valid"),
method = c("95", "mean", "sd"),
validCut = getOption("pa.validCut"),
wearThreshold = 300,
dist = c("t", "normal")
)
markingTime

Arguments

data: Data with classified wear (nonwear) status by wearingMarking.
cnts: The name of the counts column. The default is “axis1”.
markingString: Option for summarizing wear (markingString = “w”) or nonwear time (markingString = “nw”).
window: A character. It should be one of ‘trim’, ‘consecutive’, or ‘valid’.
method: A character. It should be one of ‘95’, ‘mean’ or ‘sd’.
validCut: A cutoff for the total minutes of classified monitor wear time per day to be considered as a valid monitor day.
wearThreshold: A numeric value specifying a pseudo-valid day cutoff similar to “validCut”, which is used to define a set of days to estimate the 95% CI.
dist: Option for distribution used to calculate the 95% CI.

Value

A data frame with summary information about daily counts.

Author(s)

Cole Beck <cole.beck@vumc.org>, Leena Choi <leena.choi@Vanderbilt.Edu>

Examples

data(deliveryData)
options(pa.cts = "vm")
wm <- wearingMarking(dataset = deliveryData)
markDelivery(wm)
plotData(data=wm) # days 1, 2, 10 - 15 are delivery or invalid days based on the result above
markDelivery(wm, window='valid', method='mean')
markDelivery(wm, method='mean')
markDelivery(wm, method='sd')

markingTime

Mark Days

Description

This function adds a "day" variable to the source dataset. The day is marked in numeric order, according to the timestamp variable.

Usage

markingTime(dataset, timestamp, startTime = "00:00:00", tz = "UTC")
Arguments

- **dataset**: The source dataset, in dataframe format, which needs to be marked.
- **timestamp**: The column name in the dataset that will be used as timestamp.
- **startTime**: Define the starting time of a day. It must be in the format of "hh:mm:ss".
- **tz**: Local time zone, defaults to UTC.

Value

A dataframe with an extra day marking column.

Author(s)

Zhouwen Liu <zhouwen.liu@vumc.org>

References


markPAI

*Mark Physical Activity Intensity (PAI) Level*

Description

This function adds a physical activity intensity level variable “pai” to the source dataset. The “pai” is an ordered factor variable. It will be NA for nonwear times.

Usage

```r
markPAI(
  data,
  cts = getOption("pa.cts"),
  markingString = "w",
  breaks = c(-Inf, 100, 760, 2020, Inf),
  labels = c("sedentary", "light", "moderate", "vigorous")
)
```

Arguments

- **data**: Data with classified wear (nonwear) status by `wearingMarking`.
- **cts**: The name of the counts column. The default is “axis1”.
- **markingString**: Option for summarizing wear (markingString = “w”) or nonwear time (markingString = “nw”).
- **breaks**: A numeric vector of cut-points. The default cut-points are based on Matthews *et al.* (2016).
- **labels**: A character vector labelling intensity levels.
**plotData**

value

A data frame with an additional PAI-level column.

Author(s)

Cole Beck <cole.beck@vumc.org>, Leena Choi <leena.choi@vanderbilt.edu>

References


Examples

data(dataSec)
mydata1m = dataCollapser(dataSec, TS = "TimeStamp", col = "counts", by = 60)
data1m = wearingMarking(dataset = mydata1m, perMinuteCts = 1, cts = "counts")
markPAI(data = data1m, cts = 'counts')[1:10,]

**plotData**

*Plot Accelerometer Data over Time*

Description

This function makes plot for accelerometer collected data (counts) over time for the whole monitor period, or a user specified time period or day with a midnight marking to separate monitored days.

Usage

plotData(
  data,
  day = NULL,
  start = NULL,
  end = NULL,
  cts =getOption("pa.cts"),
  TS =getOption("pa.timeStamp"),
  summary = NULL
)
Arguments

- **data**: Data with classified wear and nonwear status from `wearingMarking`.
- **day**: A part of data during a user specified day for plot.
- **start**: Define a starting time for plot.
- **end**: Define a ending time for plot.
- **cts**: The name of the counts column. The default is "axis1".
- **TS**: The column name for timestamp. The default is "TimeStamp".
- **summary**: List output of `summaryData` function.

Details

If a local time-zone is specified for `wearingMarking`, it is possible that daylight savings starts or ends during the period shown. In this case a dotted line will indicate its position and the appropriate time-zone abbreviations will be included.

Value

Plot with midnight marking.

Author(s)

Leena Choi <leena.choi@Vanderbilt.Edu>

See Also

`wearingMarking, sumVct, summaryData`

Examples

```r
data(dataSec)

mydata1m = dataCollapser(dataSec, TS = "TimeStamp", col = "counts", by = 60)

data1m = wearingMarking(dataset = mydata1m,
    frame = 90,
    perMinuteCts = 1,
    TS = "TimeStamp",
    cts = "counts",
    streamFrame = NULL,
    allowanceFrame= 2,
    newcolname = "wearing")

# change "cts" default from "axis1" to "counts"
options(pa.cts = "counts")
# plot the whole data
plotData(data=data1m)

# plot the data from 60 min to 900 min
plotData(data=data1m, start=60, end=900)
```
## plot the data for day 2
plotData(data=data1m, day=2)

## include summaryData
sumdat <- summaryData(data=data1m)
plotData(data=data1m, summary=sumdat)

## present daylight saving time change
data(deliveryData)
options(pa.cts = "vm")
wm <- wearingMarking(dataset = deliveryData, TS="TimeStamp", tz="America/Chicago")
sumdat <- summaryData(wm)
plotData(data=wm, summary = sumdat)
## valid data after delivery marking
del <- markDelivery(wm)
sumdat <- summaryData(wm, delivery = del)
plotData(data=wm, summary = sumdat)

---

### queryActigraph

**Query ActiGraph File**

#### Description

This function executes a SELECT query on an ActiGraph AGD file.

#### Usage

queryActigraph(datfile, qry)

#### Arguments

- **datfile**: An AGD file.
- **qry**: An SQL SELECT statement.

#### Details

AGD files are actually SQLite databases. This function requires the **RSQLite** package. The user is encouraged to directly interface with the database by creating a connection with the **DBI** package. This has been tested with AGD files produced with ActiLife v6.11.

#### Value

A data frame with query results.

#### Author(s)

Cole Beck <cole.beck@vumc.org>
See Also

readActigraph

Examples

```r
## Not run:
dat <- queryActigraph("actfile.agd", "SELECT * FROM data LIMIT 5")

queryActigraph("actfile.agd", "SELECT * FROM settings")

## directly interface using DBI package
con <- DBI::dbConnect(RSQLite::SQLite(), "actfile.agd")
DBI::dbListTables(con)
DBI::dbDisconnect(con)

## End(Not run)
```

---

**readActigraph**

*Read ActiGraph Accelerometer Data*

**Description**

This function reads an ActiGraph AGD file into R as a data frame. If accelerometer data are collected with three axes, it creates vector magnitude (vm). The counts at any axis or "vm" can be used to classify with wear and nonwear time using `wearingMarking`.

**Usage**

```r
readActigraph(datfile, convertTime = TRUE)
```

**Arguments**

- `datfile` An AGD file.
- `convertTime` Convert the timestamp from a character string into POSIXct.

**Details**

AGD files are SQLite databases. This function requires the `RSQLite` package.

**Value**

A data frame with accelerometer data.

**Author(s)**

Cole Beck <cole.beck@vumc.org>
### Description

This function converts accelerometer output data to a correct data format to classify wear and non-wear time using `wearingMarking`. This function can accept accelerometer output data with various epochs (for example, 1-sec, 10-sec or 1-min). If accelerometer data are collected with three axes, it creates vector magnitude (vm).

### Usage

```r
readCountsData(filename, ctPerSec, mode = 0)
```

### Arguments

- `filename` A filename of accelerometer output to be read.
- `ctPerSec` Data collection epoch. This argument tells the program the number of counting will be performed in every second. For examples: for 1-sec epoch data, set `ctPerSec = 1`; for 10-sec epoch data, set `ctPerSec = 1/10`; for 1-min epoch data, set `ctPerSec = 1/60`.
- `mode` The mode of the ActiLife dat file. Defaults to 0, and should be listed in the file header.

### Value

A data frame with the correct format (TimeStamp, counts) to be used for `wearingMarking`.
Note

Warning: It can be very slow if accelerometer data were collected with 1-sec epoch for many days.

Author(s)

Zhouwen Liu <zhouwen.liu@vumc.org>

References


See Also

wearingMarking

Examples

###############################################################################
## Read accelerometer output and convert to a correct format (TimeStamp, counts)
## Suppose "rawActigraphOutput.dat" is an Actigraph output with header as follows:
###############################################################################
## --- Data File Created By ActiGraph GT1M ActiLife v4.4.1 Firmware v7.2.0 ---
## Serial Number: LYN2B21080027
## Start Time 16:15:00
## Start Date 6/16/2010
## Epoch Period (hh:mm:ss) 00:00:01
## Download Time 09:50:23
## Download Date 6/22/2010
## Current Memory Address: 983038
## Current Battery Voltage: 4.01 Mode = 0
## ---------------------------------
###############################################################################
## This raw data with 1-sec epoch can be converted to a correct data format to
## classify wear and nonwear time using "wearingMarking" by the following code:
## Not run: mydata1s = readCountsData("rawActigraphOutput.dat", ctPerSec=1)

summaryData

`summaryData`  
*Summarize Classified Wear Time by Daily Basis*

Description

This function summarizes accelerometer data and the classified wear or nonwear time by daily basis.
Usage

```
summaryData(
  data,
  validCut = getOption("pa.validCut"),
  perMinuteCts = 1,
  markingString = "w",
  TS = getOption("pa.timeStamp"),
  cts = getOption("pa.cts"),
  delivery = NULL,
  deliveryCut = 0.5
)
```

Arguments

data       Data with classified wear (nonwear) status by `markingString`.
validCut   A cutoff for the total minutes of classified monitor wear time per day to be considered as a valid monitor day.
perMinuteCts   The number of data rows per minute. The default is 1-min epoch (perMinuteCts = 1) and we recommend to use 1-min epoch data for this summary. For examples: for data with 10-sec epoch, set perMinuteCts = 6; for data with 1-sec epoch, set perMinuteCts = 60.
markingString Option for summarizing wear (markingString = "w") or nonwear time (markingString = "nw").
TS         The column name for timestamp. The default is "TimeStamp".
cts        The name of the counts column. The default is "axis1".
delivery   data.frame. Delivery information created by `markDelivery` or `deliveryPrediction`.
deliveryCut A cutoff (probability) to consider a valid delivery date. See the `deliveryPrediction` function. The default value is 0.5.

Details

This function summarizes the total number of days, weekdays and weekend days in accelerometer data. It provides the total number of valid days, valid weekdays and valid weekend days based on a user defined cutoff for the total minutes of classified monitor wear time per day. This function also summarizes the classified wear (nonwear) time by day and by valid day, and the mean wear (nonwear) time for valid days during weekday and weekends, and for overall valid days. If mail delivery days are classified by `markDelivery`, it also summarizes the classified delivery (non-delivery) days with argument “delivery”. If “pai” column is present in the data, which can be created by `markPAI`, then physical activity intensity (PAI) level will be summarized in the output.

Value

- unit epoch for data.
- totalNumDays the total number of days in accelerometer data.
- totalNumWeekWeekend the total number of weekdays and weekend days in accelerometer data.
validCut  
a user defined cutoff for the total minutes of classified monitor wear time per day to be considered as a valid monitor day.

totalValidNumDays  
the total number of valid days based on the user defined cutoff (“validCut”) for the total minutes of wear time and the classified wear time.

totalValidNumWeekend  
the total number of valid weekdays and valid weekend days based on the user defined cutoff (“validCut”) for the total minutes of classified monitor wear time per day.

wearTimeByDay  
the classified total wear (nonwear) time by day.
deliveryDays  
marked delivery days.
validWearTimeByDay  
the classified total wear (nonwear) time by valid day.
meanWearTimeValidDays  
the mean wear (nonwear) time for valid days during weekdays and weekends.
meanWearTimeOverallValidDays  
the mean wear (nonwear) time for overall valid days.
dayInfo  
information about wear time and mean counts for each day.
intensity  
optional output depending on “pai” column in the data; the total time in hours of physical activity intensity by day.
meanValidIntensity  
optional output depending on “pai” column in the data; the mean physical activity intensity (PAI) level for valid days.

Author(s)
Cole Beck <cole.beck@vumc.org>, Leena Choi <leena.choi@Vanderbilt.Edu>

References

See Also
wearingMarking, sumVct, markPAI, markDelivery

Examples

data(dataSec)

mydata1m = dataCollapser(dataSec, TS = "TimeStamp", col = "counts", by = 60)

data1m = wearingMarking(dataset = mydata1m,
        frame = 90,
        perMinuteCts = 1,
        TS = "TimeStamp",
        cts = "counts",
        ...
sumVct

streamFrame = NULL,
allowanceFrame= 2,
newcolname = "wearing")

summaryData(data=data1m, validCut=600, perMinuteCts=1, markingString = "w", cts = "counts")

data(deliveryData)
options(pa.cts = "vm")
wm <- wearingMarking(dataset = deliveryData)
bd <- markDelivery(wm)
pdd <- deliveryPred(wm)
summaryData(wm, delivery = bd)
summaryData(wm, delivery = pdd)

pai.data <- markPAI(data = wm)
bd <- markDelivery(pai.data)
summaryData(pai.data, delivery = bd)

sumVct

**Summarize Wear and Nonwear Time Interval**

**Description**

This function summarizes the classified wear (nonwear) time by interval basis from the epoch-by-epoch classified wear (nonwear) status classified by `wearingMarking`.

**Usage**

```r
sumVct(  
datavct,  
wearing = "wearing",  
TS = getOption("pa.timeStamp"),  
markingString = "w",  
by = "days",  
id = NULL  )
```

**Arguments**

- `datavct`: Data with classified wear (nonwear) status classified by `wearingMarking`.
- `wearing`: The column name for classified wear and nonwear status. The default is "wearing".
- `TS`: The column name for timestamp. The default is "TimeStamp".
- `markingString`: Option for summarizing wear (markingString="w") or nonwear time interval (markingString="nw").
- `by`: A sequence of days for classified wear (nonwear) time intervals.
- `id`: Optional output for subject identification or file name.
wearingMarking

Value
The summary data for wear and nonwear time intervals.

Author(s)
Leena Choi <leena.choi@Vanderbilt.edu>, Cole Beck <cole.beck@vumc.org>, Zhouwen Liu <zhouwen.liu@vumc.org>, Charles E. Matthews <Charles.Matthews2@nih.gov>, and Maciej S. Buchowski <maciej.buchowski@Vanderbilt.edu>

References

See Also
wearingMarking, summaryData

Examples

data(dataSec)
mydata1m = dataCollapser(dataSec, TS = “TimeStamp”, col = ”counts”, by = 60)
data1m = wearingMarking(dataset = mydata1m,
                        frame = 90,
                        perMinuteCts = 1,
                        TS = ”TimeStamp”,
                        cts = ”counts”,
                        streamFrame = NULL,
                        allowanceFrame= 2,
                        newcolname = ”wearing”)

sumVct(data1m, id= ”sdata1m”)
sumVct(data1m, id= ”sdata1m”, markingString = ”nw”)

wearingMarking  Classify Wear and Nonwear Time for Accelerometer Data

Description
This function classifies wear and nonwear time status for accelerometer data by epoch-by-epoch basis.
wearingMarking

Usage

wearingMarking(
  dataset,
  frame = 90,
  perMinuteCts = 60,
  TS = getOption("pa.timeStamp"),
  cts = getOption("pa.cts"),
  streamFrame = NULL,
  allowanceFrame = 2,
  newcolname = "wearing",
  getMinuteMarking = FALSE,
  dayStart = "00:00:00",
  tz = "UTC",
  ...
)

Arguments

dataset The source dataset, in dataframe format, which needs to be marked.
frame The size of time interval to be considered; Window 1 described in Choi et al. (2011). The default is 90.
perMinuteCts The number of data rows per minute. The default is 1-sec epoch (perMinuteCts = 60). For examples: for data with 10-sec epoch, set perMinuteCts = 6; for data with 1-min epoch, set perMinuteCts = 1.
TS The column name for timestamp. The default is “TimeStamp”.
cts The column name for counts. The default is “axis1”.
streamFrame The size of time interval that the program will look back or forward if activity is detected; Window 2 described in Choi et al. (2011). The default is the half of the frame.
allowanceFrame The size of time interval that zero counts are allowed; the artifactual movement interval described in Choi et al. (2011). The default is 2.
newcolname The column name for classified wear and nonwear status. The default is “wearing”. After the data is processed, a new field will be added to the original dataframe. This new field is an indicator for the wearing (“w”) or nonwearing (“nw”).
getMinuteMarking Return minute data with wear and nonwear classification. If the source is not a minute dataset, the function will collapse it into minute data. The default is FALSE.
dayStart Define the starting time of day. The default is the midnight, "00:00:00". It must be in the format of "hh:mm:ss".
 tz Local time zone, defaults to UTC.
... Parameter settings that will be used in dataCollapser function.
Details

A detailed description of the algorithm implemented in this function is described in Choi et al. (2011).

Value

A data frame with the column for wear and nonwear classification indicator by epoch-by-epoch basis.

Note

Warning: It will be very slow if accelerometer data with 1-sec epoch for many days are directly classified. We recommend to collapse a dataset with 1-sec epoch to 1-min epoch data using dataCollapser and then classify wear and nonwear status using a dataset with a larger epoch.

Author(s)

Leena Choi <leena.choi@Vanderbilt.Edu>, Cole Beck <cole.beck@vumc.org>, Zhouwen Liu <zhouwen.liu@vumc.org>, Charles E. Matthews <Charles.Matthews2@nih.gov>, and Maciej S. Buchowski <maciej.buchowski@Vanderbilt.Edu>

References


See Also

readCountsData, sumVct

Examples

data(dataSec)

## mark data with 1-min epoch
mydata1m = dataCollapser(dataSec, TS = "TimeStamp", col = "counts", by = 60)

data1m = wearingMarking(dataset = mydata1m,
frame = 90,
perMinuteCts = 1,
TS = "TimeStamp",
cts = "counts",
streamFrame = NULL,
allowanceFrame= 2,
newcolname = "wearing")

sumVct(data1m, id="dataid")

## mark data with 1-sec epoch
## Not run:
data1s = wearingMarking(dataset = dataSec,
wearingMarking

frame = 90,
perMinuteCts = 60,
TS = "TimeStamp",
cts = "counts",
streamFrame = NULL,
allowanceFrame = 2,
newcolname = "wearing",
getMinuteMarking = FALSE)

sumVct(datalst, id="dataid")
sumVct(datalst, id="dataid", markingString = "nw")

## End(Not run)
Index

* accelerometer data process
  PhysicalActivity-package, 2

* datasets
  dataSec, 4
deliveryData, 5

* mail delivery day classification
  PhysicalActivity-package, 2

* wear and nonwear classification
  PhysicalActivity-package, 2

dataCollapsr, 3, 23, 24
dataSec, 4
deliveryData, 5
deliveryFeatures, 6, 7–9
deliveryPred, 6, 7, 9, 10
deliveryPrediction, 7, 8, 19
deliveryPreprocess, 7, 9
markDelivery, 2, 10, 19, 20
markingTime, 11
markPAI, 2, 12, 19, 20

PhysicalActivity
  (PhysicalActivity-package), 2
PhysicalActivity-package, 2
plotData, 13
queryActigraph, 2, 15, 17
readActigraph, 2, 16, 16
readCountsData, 17, 24
summaryData, 14, 18, 22
sumVct, 14, 20, 21, 24
wearingMarking, 2, 11, 12, 14, 16–22, 22