Package 'GGIR'

August 8, 2016

Type Package

Title Raw Accelerometer Data Analysis

Version 1.2-10

Date 2016-08-05

Maintainer Vincent T van Hees <vincentvanhees@gmail.com>

Description A tool to process and analyse data collected with wearable raw acceleration sensors. The package has been developed and tested for binary data from GENEActiv and GENEA devices and .csv-export data from Actigraph devices. These devices are currently widely used in research on human daily physical activity.

License LGPL (>= 2.0, < 3)

Suggests MASS, signal, zoo, mmap, bitops, matlab, GENEAread, tuneR

Depends stats, utils, R (>= 2.10)

NeedsCompilation no

Author Vincent T van Hees [aut, cre],
Zhou Fang [ctb],
Jing Hua Zhao [ctb],
Severine Sabia [ctb]

R topics documented:

GGIR-package 2
data.calibrate 4
data.getmeta 5
data.inspectfile 5
g.analyse 6
g.binread 10
g.calibrate 11
g.getmeta 13
g.impute 16
g.inspectfile 17
g.part1 18
g.part2 21

g.part3 24
g.part4 25
g.plot 29
g.shell.GGIR 30

Index **34**

GGIR-package *A package to process multi-day raw accelerometer data*

Description

GGIR is an R-package to process multi-day raw accelerometer data. It was developed in the context of research on human daily physical activity with wearable tri-axial acceleration sensors. The term raw accelerometry refers to data being expressed in m/s2 or gravitational acceleration as opposed to the previous generation accelerometers which stored only processed summary measures.

The package has been developed with and for the accelerometer brands Genea and Geneactive. Additionally, it should work for csv data from Geneactiv and Actigraph accelerometer data. Although, I have tested this less thoroughly compared with the binary data formats from Genea and Geneactiv.
Note for Actigraph users: please do not export timestamps to the csv-file as this causes memory issues. To cope with the absense of timestamps the code will re-caculate timestamps from the sample frequency and the start time and date as presented in the file header.

Function g.inspectfile assessess to which monitor brand the file belongs and extracts the file header; function g.calibrate helps to investigate calibration error based on free-living data and proposes correction factors; function g.getmeta extracts the signal features; g.impute takes that information, identifies unreliable signal sections (e.g. monitor not worn or signal clips near its extreme) and replaces these sections by imputed values; and finally g.analyse takes the output from all the functions, runs a basic descriptive analysis and then summarises the output both per measurement and per day of measurement.

To enhance the feasibility of using these individual functions I am providing a couple of shell functions to ease implementing the above functions in study data by less experienced R-users. Here, the main shell function is g.shell.GGIR and allows for automating the full analysis of a dataset including all necessary calls to the functions above. Function g.shell.GGIR relies on functions g.part1 and g.part2 also part of this package. In summary, the user is expected to specify the location of the acceleerometer data and the desired output folder. Next, data is loaded and pre- processed with g.getmeta and g.calibrate. Next, the output is converted to a conveniently portable .RData-format away from the R workspace. Next, these .RData files are used as input for g.part2.

Note that g.part1 generates a folder structure to help the user keep track of various output files and milestone data. The folder structure entails: One master folder with a name output_xx where xx is equal to the name of the original data folder. Inside the output_xx folder there will be one folder named meta including all the milestone and a folder results with all the results. Inside the meta folder the following subfolders are created: basic, ms2.out, ms3out, and ms4out for respectively g.part1, g.part2, g.part3, and g.part4 milestone data.

The reason why g.part1 and g.part2 are not merged as one generic shell function is because g.part1 takes much longer to run and involves only minor decisions of interest to the movement scientist. Function g.part2 on the other hand is relatively fast and comes with all the decisions that directly impact on the variables that are of interest to the movement scientist. Therefore, the user may want to run g.part1 overnight or on a computing cluster, while g.part2 can then be the main playing ground for the movement scientist. So, function g.shell.GGIR basically is the central point for operating both g.part1 and g.part2 and most users should not really need to interact with g.part1 or g.part2 directly. More recently I expanded the package with g.part3 and g.part4 which provide functionality for estimating sleep and sustained inactivity bouts.

If you want to use this package for a different data format (e.g. from a different accelerometer brand) then please provide me with: the R-code to read the data and example files for testing purposes.

Please note that there is google discussion group for this package (link below).

You can thank me for sharing the code in this package and for developing it as a generic purpose tool by citing the package name and by citing the supporting publications in your own scientific journal/conference publications.

Details

Package:	GGIR
Type:	Package
Version:	1.2-10
Date:	2016-8-05
License:	LGPL (>= 2.0, < 3)
Discussion group:	https://groups.google.com/forum/#!forum/rpackageggir

Author(s)

- Vincent T van Hees <vincentvanhees@gmail.com>
- Zhou Fang co-developed function g.calibrate
- Jing Hua Zhao <jinghua.zhao@mrc-epid.cam.ac.uk> co-developed function g.binread
- Severine Sabia tested and provided feedback on various functions

References

- van Hees VT, Gorzelniak L, Dean Leon EC, Eder M, Pias M, et al. (2013) Separating Movement and Gravity Components in an Acceleration Signal and Implications for the Assessment of Human Daily Physical Activity. PLoS ONE 8(4): e61691. doi:10.1371/journal.pone.0061691
- van Hees VT, Fang Z, Langford J, Assah F, Mohammad A, da Silva IC, Trenell MI, White T, Wareham NJ, Brage S. Auto-calibration of accelerometer data for free-living physical activity

assessment using local gravity and temperature: an evaluation on four continents. J Appl Physiol (1985). 2014 Aug 7

- van Hees VT, Sabia S, et al. (2015) A novel, open access method to assess sleep duration using a wrist-worn accelerometer, PLoS ONE, November 2015

Examples

```
## Not run:
  #inspect file:
  I = g.inspectfile(datafile)

  #autocalibration:
  C = g.calibrate(datafile)

  #get meta-data:
  M = g.getmeta(datafile)

## End(Not run)
data(data.getmeta)
data(data.inspectfile)
data(data.calibrate)

#impute meta-data:
IMP = g.impute(M = data.getmeta, I = data.inspectfile)
#analyse and produce summary:
A = g.analyse(I = data.inspectfile, C = data.calibrate, M = data.getmeta, IMP)
#plot data
g.plot(IMP, M = data.getmeta, I = data.inspectfile, durplot=4)
```

---

data.calibrate *Example output from g.calibrate*

Description

data.calibrate is example output from g.calibrate

Usage

```
data(data.calibrate)
```

Format

The format is: chr "data.calibrate"

Source

The data was collected on one individual for testing purposes

Examples

```
data(data.calibrate)
```

`data.getmeta` *Example output from g.getmeta*

Description

data.getmeta is example output from g.getmeta

Usage

```
data(data.getmeta)
```

Format

The format is: chr "data.getmeta"

Source

The data was collected on one individual for testing purposes

Examples

```
data(data.getmeta)
```

`data.inspectfile` *Example output from g.inspectfile*

Description

data.inspectfile is example output from g.inspectfile

Usage

```
data(data.inspectfile)
```

Format

The format is: chr "data.inspectfile"

Source

The data was collected on one individual for testing purposes

Examples

```
data(data.inspectfile)
```

g.analyse *function to analsyse meta-data generated by g.getmeta and g.impute*

Description

Analyses the output from other functions within the packages to generate a basic descriptive summary for each accelerometer data file. Analyses include: Average acceleration per day, per measurement, L5M5 analyses (assessment of the five hours with lowest acceleration and with highest acceleration). Further, the traditionally popular variable MVPA is automatically extracted in six variants: without bout criteria in combination with epoch = epoch length as defined in g.getmeta (first value of the input argument windowsizes), 1 minute, and 5 minutes, and for bout durations 1 minute, 5 minutes or 10 minutes in combination with the epoch length as defined in g.getmeta.

Usage

```
g.analyse(I, C, M, IMP, qlevels = c(), qwindow = c(0, 24), quantiletype = 7,
L5M5window = c(0, 24), M5L5res = 10, includedaycrit = 16, ilevels = c(),
winhr = 5, idloc = 1,snloc=1,mvpathreshold = c(),boutcriter=c(),mvpadur=c(1,5,10),
selectdaysfile=c(),window.summary.size=10,dayborder=0,mvpa.2014 = FALSE,
closedbout=FALSE,desiredtz = c())
```

Arguments

I	the output from function g.inspectfile
C	the output from function g.calibrate
M	the output from function g.getmeta
IMP	the output from function g.impute
qlevels	array of percentiles for which value needs to be extracted. These need to be expressed as a fraction of 1, e.g. c(0.1, 0.5, 0.75). There is no limit to the number of percentiles. If left empty then percentiles will not be extracted. Distribution will be derived from short epoch metric data, see g.getmeta.
qwindow	start and end time, in 24 hour clock hours, over which distribution in metric values need to be extracted. Default value = c(0,24) will consider all 24 hours.
quantiletype	type of quantile function to use (default recommended). For details, see quantile function in STATS package
L5M5window	start and end time, in 24 hour clock hours, over which L5M5 needs to be calculated.
M5L5res	resoltion of L5 and M5 analysis in minutes (default: 10 minutes)
includedaycrit	minimum required number of valid hours in day specific analysis (NOTE: there is no minimum required number of hours per day in the summary of an entire measurement, every available hour is used to make the best possible inference on average metric value per average day)
ilevels	Levels for acceleration value frequency distribution in mg, e.g. c(0,100,200) There is no constriction to the number of levels.

winhr	window size in hours of L5 and M5 analysis (dedault = 5 hours)
idloc	If value = 1 (default) the code assumes that ID number is stored in the obvious header field. If value = 2 the code uses the character string preceding the character '_' in the filename as the ID number
snloc	If value = 1 (default) the code assumes that device serial number is stored in the obvious header field. If value = 2 the code uses the character string between the first and second character '_' in the filename as the serial number
mvpathreshold	Threshold for MVPA estimation. This can be a single number or an array of numbers, e.g. c(100,120). In the later case the code will estimate MVPA seperately for each threshold. If this variable is left blank c() then MVPA is not estimated
boutcriter	The variable boutcriter is a number between 0 and 1 and defines what fraction of a bout needs to be above the mvpathreshold
mvpadur	default = c(1,5,10). Three bout duration for which MVPA will be calculated
selectdaysfile	Functionality designed for the London Centre of Longidutinal studies. Csv file holding the relation between device serial numbers and measurement days of interest.
dayborder	Hour at which days start and end (default = 0), value = 4 would mean 4am
window.summary.size	Functionality designed for the London Centre of Longidutinal studies. Size in minutes of the summary window
mvpa.2014	If TRUE use the MVPA bout definition as has been available since 2014. The algorithm looks for 10 minute windows in which more than 80 percent of the epochs are above mvpathreshold, and then counts the entire window as mvpa. The new bout definition (mvpa.2014 = FALSE) looks for a group or groups of epochs with a value above mvpathreshold that span a time window of at least mvpadur minutes in which more than boutcriter percent of the epochs are above the threshold. The motivation for the old defition was: A person who spends 10 minutes in MVPA with a 2 minute break in the middle is equally active as a person who spends 8 minutes in MVPA without taking a break. Therefore, both should be counted equal and counted as 10 minute MVPA bout. The motivation for the new definition is: not counting breaks towards MVPA simplifies interpretation and still counts the two persons in the example as each others equal.
closedbout	If TRUE then count breaks in a bout towards the bout duration. If FALSE then only count time spent above the threshold towards the bout duration.
desiredtz	see g.getmeta

Details

The value summary is a dataframe and comes with the following variables:

- ID Participant id extracted from file header
- device_sn Device serial number extracted from file header
- dodylocation Body location extracted from file header

- filename Name of the accelerometer file
- start_time Timestamp when experiment started
- startday Name of day when experiment started
- samplefreqSample frequency (Hz)
- device Name of the device brand, e.g. Geneactiv
- clipping_score Fraction of 15 minute windows per file for which the acceleration in one of the three axis was close to the maximum for at least 80 percent of the time. This should be 0
- meas_dur_dys Measurement duration (days)
- complete_24hrcycle Fraction of 15 minute windows per 24 hours for which valid data is available at any day of the measurement
- meas_dur_def_proto_day Measurement duration (days) minus the hours that are ignored at the beginning and end of the measurement motived by protocol design
- wear_dur_def_proto_day Measurement duration according to protocol (days) minus invalid time periods
- calib_err Estimated based on all non-movement periods in the measurement after applying the autocalibration
- calib_status Summary statement about the status of the calibration error minimisation
- ENMO ENMO is the main summary measure of acceleration. The value presented is the average ENMO over all the available data normalised per 24 hour cycles, with invalid data imputed by the average at similar timepoints on different days of the week. In addition to ENMO it is possible to extract other acceleration metrics (i.e. BFEN, HFEN, HFENplus)
- pX_ENMO_mg_0-24h This variable represents the Xth percentile in the distribution of short epoch acceleration values of the average day within the time interval as specified.
- L5hr_ENMO_mg_0-24 Starting time in hours of the least active five* hours within the time interval as specified (* window size is modifiable in g.getmeta)
- L5_ENMO_mg Average acceleration over L5
- M5hr_ENMO_mg_0-24 Starting time in hours of the most active five* hours in the day within the time interval as specified (* window size is modifiable in g.getmeta)
- M5_ENMO_mg_0-24 Average acceleration over M5
- Accelerationa 1am-6am value of ENMO (mg) Average acceleration between 1am and 6am
- N valid WEdays Number of valid weekend days
- N valid WDdays Number of valid week days
- AD_... The variable ... was calculated per day and then averaged over all the available days
- WE_... The variable ... was calculated per day and then averaged over weekend days only
- WD_... The variable ... was calculated per day and then averaged over week days only
- WWE_... The variable ... was calculated per day and then averaged over weekend days. Double weekend days are averaged This is only relevant for experiments that last for more than seven days
- WWD_... The variable ... was calculated per day and then averaged over week days. Double weekend days were averaged. This is only relevant for experiments that last for more than seven days)

- `..._MVPA_E5S_B1M80_T100` MVPA calculated based on 5 second epoch setting bout duration 1 Minute and inclusion criterion of more than 80 percent. This is only done for metric ENMO at the moment, and only if mvpathreshold is not left blank
- `..._mean_ENMO...` ENMO or other metric was first calcualte per day and then average according to AD, WD, WWE, WWD
- `data exclusion strategy` A log of the decision made when calling g.impute: value=1 mean ignore specific hours; value=2 mean ignore all data before the first midnight and after the last midnight
- `n hours ignored at the start of the measurement (if strategy = 1)` A log of the decision made when calling g.impute
- `n hours ignored at the end of the measurement (if strategy = 1)` A log of the decision made when calling g.impute
- `n days of measurement after which data is ignored (if strategy = 1)` A log of the decision made when calling g.impute

The value `daysummary` is a dataframe and comes with the following variables:

- `ID` Participant id extracted from file header
- `filename` File name
- `calender_date` Calender data
- `bodylocation` Body location (if known)
- `N valid hours` Number of hours with valid data
- `N hours` Number of hours of measurement
- `Day of the week`Day of the week
- `Day of measurement` Day number relative to start of the measurement
- `L5_ENMO_mg_0-24h` Magnitude of average acceleration during the least active five hours calculated with metric ENMO. Within the time window as specified
- `L5hr_ENMO_mg_0-24h` Starting hour of L5 on a scale from 0 to 24, where 14.5 means 14:30. Within the time window as specified
- `M5_ENMO_mg_0-24h` Magnitude of average acceleration during the most active five hours calculated with metric ENMO. Within the time window as specified
- `M5hr_ENMO_mg_0-24h` Starting hour of M5 on a scale from 0 to 24, where 14.5 means 14:30. Within the time window as specified
- `mean_ENMO_mg_1-6am` Mean acceleration between 1am and 6am
- `mean_ENMO_mg_24hr` Mean acceleration over 24 hour period
- `pX_ENMO_mg_0-24h` Percentile in the short epoch distribution with invalid data imputed. Within the time window as specified
- `[A,B)_ENMO_mg_0-24h` Time spent in minutes between (and including) acceleration value A in mg and (excluding) acceleration value B in mg. This is only done for metric ENMO at the moment, and only done if ilevels is not left blank
- `MVPA_E5S_B1M80_T100` MVPA calculated based on 5 second epoch setting bout duration 1 Minute and inclusion criterion of more than 80 percent. This is only done for metric ENMO at the moment, and only if mvpathreshold is not left blank

Value

summary	summary for the file that was analysed (see details)
daysummary	summary per day for the file that was analysed (see details)

Author(s)

Vincent T van Hees <vincentvanhees@gmail.com>

Examples

```
data(data.getmeta)
data(data.inspectfile)
data(data.calibrate)
## Not run:
  #inspect file:
  I = g.inspectfile(datafile)

  #autocalibration:
  C = g.calibrate(datafile)

  #get meta-data:
  M = g.getmeta(datafile, desiredtz = "Europe/London", windowsizes = c(5, 900, 3600),
  daylimit = FALSE, offset = c(0, 0, 0), scale = c(1, 1, 1), tempoffset = c(0, 0, 0))

## End(Not run)


#impute meta-data:
IMP = g.impute(M = data.getmeta, I = data.inspectfile)

#analyse and produce summary:
A = g.analyse(I = data.inspectfile, C = data.calibrate, M = data.getmeta, IMP)
```

---

g.binread

function to read binary files as produced by the accelerometer named 'Genea', not to be confused with the 'GENEActiv' (see package GENEAread for this)

Description

For reading the binary data as collected with a Genea accelerometer (Unilever Discover, UK). For reading GENEActive binary data, see package GENEAread.

Usage

```
g.binread(binfile, start = 0, end = 0)
```

Arguments

binfile	filename (required)
start	start point for reading data, this can either be a timestamp "year-month-day hr:min:sec" or a page number (optional)
end	end point for reading data, this can either be a timestamp "year-month-day hr:min:sec" or a page number (optional)

Details

If only start is defined then g.binread will read all data beyond start until the end of the file is reached

Value

rawxyz	matrix with raw x, y, and, z acceleration values
header	file header
timestamps1	timestamps for rawxyz in seconds since 1970-01-01 00:00
timestamps2	timestamps for rawxyz in day time format
batt.voltage	matrix with battery voltage and corresponding timestamps

Author(s)

Vincent T van Hees <vincentvanhees@gmail.com> Jing Hua Zhao <jinghua.zhao@mrc-epid.cam.ac.uk>

---

g.calibrate *function to estimate calibration error and make recommendation for addressing it*

Description

Function starts by identifying ten second windows of non-movement. Next, the average acceleration per axis per window is used to estimate calibration error (offset and scaling) per axis. The function provides recommended correction factors to address the calibration error and a summary of the callibration procedure.

Usage

```
g.calibrate(datafile, use.temp = TRUE, spherecrit = 0.3, minloadcrit = 72,
printsummary = TRUE,chunksize=c(),windowsizes=c(5,900,3600),selectdaysfile=c(),
dayborder=0)
```

Arguments

datafile	name of accelerometer file
use.temp	use temperature sensor data if available (Geneactive only)
spherecrit	the minimum required acceleration value (in g) on both sides of 0 g for each axis. Used to judge whether the sphere is sufficiently populated
minloadcrit	the minimum number of hours the code needs to read for the autocalibration procedure to be effective (only sensitive to multitudes of 12 hrs, other values will be ceiled). After loading these hours only extra data is loaded if calibration error has not been reduced to under 0.01 g.
printsummary	if TRUE will print a summary when done
chunksize	number between 0.2 and 1 to specificy the size of chunks to be loaded as a fraction of a 12 hour period, e.g. 0.5 equals 6 hour chunks. The default is 1 (12 hrs). For machines with less than 4Gb of RAM memory a value below 1 is recommended.
windowsizes	see g.getmeta
selectdaysfile	see g.part1
dayborder	see g.part1

Value

scale	scaling correction values, e.g. c(1,1,1)
offset	offset correction values, e.g. c(0,0,0)
tempoffset	correction values related to temperature, e.g. c(0,0,0)
cal.error.start	absolute difference between Euclidean norm during all non-movement windows and 1 g before autocalibration
cal.error.end	absolute difference between Euclidean norm during all non-movement windows and 1 g after autocalibration
spheredata	average, standard deviation, Euclidean norm and temperature (if available) for all ten second non-movement windows as used for the autocalibration procedure
npoints	number of 10 second no-movement windows used to populate the sphere
nhoursused	number of hours of measurement data scanned to find the ten second time windows with no movement
meantempcal	mean temperature corresponding to the data as used for autocalibration. Only applies to data collected with GENEActiv monitor.

Author(s)

Vincent T van Hees <vincentvanhees@gmail.com> Zhou Fang

References

- van Hees VT, Fang Z, Langford J, Assah F, Mohammad A, da Silva IC, Trenell MI, White T, Wareham NJ, Brage S. Auto-calibration of accelerometer data for free-living physical activity assessment using local gravity and temperature: an evaluation on four continents. J Appl Physiol (1985). 2014 Aug 7

Examples

```
## Not run:
datafile = "C:/myfolder/testfile.bin"

#Apply autocalibration:
C = g.calibrate(datafile)
print(C$scale)
print(C$offset)

## End(Not run)
```

`g.getmeta` *function to extract meta-data (features) from data in accelerometer file*

Description

Reads a accelerometer file in blocks, extracts various features and stores average feature value per short or long epoch. Acceleration and angle metrics are stored at short epoch length. The non-wear indication score, the clipping score, temperature (if available), light (if available), and Euclidean norm are stored at long epoch length. The function has been designed and thoroughly tested with accelerometer files from GENEA and GENEActiv. Further, the function should be able to cope with csv-format data procuded by GENEActiv and Actigraph

Usage

```
g.getmeta(datafile, desiredtz = c(),
windowsizes = c(5, 900, 3600),daylimit = FALSE,
offset = c(0,0,0), scale = c(1,1,1),
tempoffset = c(0,0,0),do.bfen = FALSE, do.enmo = TRUE,
do.lfenmo = FALSE, do.en = FALSE,
do.hfen = FALSE, do.hfenplus = FALSE,
do.teLindert2013=FALSE,do.anglex=FALSE,do.angley=FALSE,do.anglez=FALSE,
do.roll_med_acc_x=FALSE,do.roll_med_acc_y=FALSE,do.roll_med_acc_z=FALSE,
do.dev_roll_med_acc_x=FALSE,do.dev_roll_med_acc_y=FALSE,do.dev_roll_med_acc_z=FALSE,
do.enmoa = FALSE,lb = 0.2,hb = 15, n = 4,meantempcal=c(),chunksize=c(),
selectdaysfile=c(),dayborder=0,...)
```

Arguments

`datafile`	name of accelerometer file
`desiredtz`	desired timezone: see also http://en.wikipedia.org/wiki/Zone.tab
`windowsizes`	Three values to indicate the lengths of the windows as in c(window1,window2,window3): window1 is the short epoch length in seconds and by default 5 this is the time window over which acceleration and angle metrics are calculated, window2 is the long epoch length in seconds for which non-wear and signal clipping are defined, default 900. However, window3 is the window length of data used for non-wear detection and by default 3600 seconds. So, when window3 is larger

than window2 we use overlapping windows, while if window2 equals window3 non-wear periods are assessed by non-overlapping windows.

`daylimit` number of days to limit (roughly), if set to FALSE no daylimit will be applied

`offset` offset correction value per axis, usage: value = scale(value,center = -offset, scale = 1/scale)

`scale` scaling correction value per axis, usage: value = scale(value,center = -offset, scale = 1/scale)

`tempoffset` temperature offset correction value per axis, usage: value = scale(value,center = -offset, scale = 1/scale) + scale(temperature, center = rep(averagetemperate,3), scale = 1/tempoffset)

`do.bfen` if TRUE, calculate metric BFEN with band-pass filter configuration set by `lb` and `hb`

`do.enmo` if TRUE (default), calculate metric ENMO with negative values rounded to zero

`do.lfenmo` if TRUE, calculate metric LFENMO with low-pass filter configuration set by `hb`

`do.en` if TRUE, calculate metric EN

`do.hfen` if TRUE, calculate metric HFEN with low-pass filter configuration set by `hb`

`do.hfenplus` if TRUE, calculate metric HFENplus with band-pass filter configuration set by `lb` and `hb`

`do.teLindert2013` if TRUE, calculate the 5 sec epoch Actiwatch count replication as described by te Lindert et al 2013 in the journal SLEEP (volume 36, issue 5, page 781)

`do.anglex` if TRUE, calculate the angle of the x-axis relative to the horizontal plane (degrees) utilizing all three axes

`do.angley` if TRUE, calculate the angle of the y-axis relative to the horizontal plane (degrees) utilizing all three axes

`do.anglez` if TRUE, calculate the angle of the z-axis relative to the horizontal plane (degrees) utilizing all three axes

`do.enmoa` if TRUE (default), calculate metric ENMOa which is equal to metric ENMO but with the absolute taken from the Euclidean norm minus one.

`do.roll_med_acc_x` if TRUE, calculate rolling median for the x axis

`do.roll_med_acc_y` if TRUE, calculate rolling median for the y axis

`do.roll_med_acc_z` if TRUE, calculate rolling median for the z axis

`do.dev_roll_med_acc_x` if TRUE, calculate deviations from rolling median for the x axis

`do.dev_roll_med_acc_y` if TRUE, calculate deviations from rolling median for the y axis

`do.dev_roll_med_acc_z` if TRUE, calculate deviations from rolling median for the z axis

`lb` lower boundary of the frequency filter (in Hertz)

hb	upper boundary of the frequency filter (in Hertz)
n	order of the frequency filter
meantempcal	mean temperature corresponding to the data as used for autocalibration. If autocalibration is not done or if temperature was not available then leave blank (default)
chunksize	number between 0.2 and 1 to specificy the size of chunks to be loaded as a fraction of a 24 hour period, e.g. 0.5 equals 12 hour chunks. The default is 1 (24 hrs). For machines with less than 4Gb of RAM memory a value below 1 is recommended.
selectdaysfile	see g.part1
dayborder	see g.part1
...	Please ignore. Only used by the code internally when called from within g.part1 with selectdaysfile specific.

Value

metalong	dataframe with long epoch meta-data: EN, non-wear score, clipping score, temperature
metashort	dataframe with short epoch meta-data: timestamp and metric
tooshort	indicator of whether file was too short for processing (TRUE or FALSE)
corrupt	indicator of whether file was considered corrupt (TRUE or FALSE)

Author(s)

Vincent T van Hees <vincentvanhees@gmail.com>

References

van Hees VT, Gorzelniak L, Dean Leon EC, Eder M, Pias M, et al. (2013) Separating Movement and Gravity Components in an Acceleration Signal and Implications for the Assessment of Human Daily Physical Activity. PLoS ONE 8(4): e61691. doi:10.1371/journal.pone.0061691

Examples

```
## Not run:
datafile = "C:/myfolder/testfile.bin"

#Extract meta-data:
M = g.getmeta(datafile)

#Inspect first couple of rows of long epoch length meta data:
print(M$metalong[1:5,])

#Inspect first couple of rows of short epoch length meta data:
print(M$metalong[1:5,])

## End(Not run)
```

g.impute *function to identify invalid periods in the meta-data as generated by g.getmeta and to impute these invalid periods with the average of similar timepoints on other days of the measurement*

Description

Functions takes the output from g.getmeta and information about the study protocol to label impute invalid time segments in the data.

Usage

```
g.impute(M, I, strategy = 1, hrs.del.start = 0, hrs.del.end = 0, maxdur = 0,
ndayswindow = 7,desiredtz="Europe/London")
```

Arguments

M	output from g.getmeta
I	output from g.inspectfile
strategy	how to deal with knowledge about study protocol. value = 1 means select data based on hrs.del.start, hrs.del.end, and maxdur. Value = 2 makes that only the data between the first midnight and the last midnight is used for imputation. Value = 3 only selects the most active X days in the files. X is specified by argument ndayswindow
hrs.del.start	how many HOURS after start of experiment did wearing of monitor start?
hrs.del.end	how many HOURS before the end of the experiment did wearing of monitor definitely end?
maxdur	How many DAYS after start of experiment did experiment definitely stop? (set to zero if unknown = default)
ndayswindow	If strategy is set to 3 then this is the size of the window as a number of days
desiredtz	see g.getmeta

Value

metashort	imputed short epoch variables
rout	matrix to clarify when data was imputed for each long epoch time window and the reason for imputation. Value = 1 indicates imputation. Columns 1 = monitor non wear, column 2 = clipping, column 3 = additional nonwear, column 4 = protocol based exclusion and column5 = sum of column 1,2,3 and 4.
averageday	matrix with n columns for n metrics values and m rows for m short epoch time windows in an average 24 hours period

Author(s)

Vincent T van Hees <vincentvanhees@gmail.com>

Examples

```
## Not run:
  #inspect file:
  I = g.inspectfile(datafile)

  #autocalibration:
  C = g.calibrate(datafile)

  #get meta-data:
  M = g.getmeta(datafile)

## End(Not run)

data(data.getmeta)
data(data.inspectfile)

#impute meta-data:
IMP = g.impute(M=data.getmeta, I=data.inspectfile)
```

`g.inspectfile` *function to inspect accelerometer file for brand, sample frequency and header*

Description

Inspects accelerometer file for key information, including: monitor brand, sample frequency and file header

Usage

```
g.inspectfile(datafile)
```

Arguments

`datafile`	name of data file

Value

`header`	fileheader
`monn`	monitor name (genea, geneactive)
`monc`	monitor brand code (1 = genea; 2 = geneactive, 3 = actigraph)
`dformn`	data format (bin, csv)
`dformc`	data format code (1 = bin, 2 = csv)
`sf`	samplefrequency in Hertz
`filename`	filename

Author(s)

Vincent T van Hees <vincentvanhees@gmail.com>

`g.part1` *function to load and pre-process acceleration files*

Description

Calls function g.getmeta and g.calibrate, and converts the output to .RData-format which will be the input for g.part2. Here, the function generates a folder structure to keep track of various output files. The reason why these g.part1 and g.part2 are not merged as one generic shell function is because g.part1 takes much longer to and involves only minor decisions of interest to the movement scientist. Function g.part2 on the other hand is relatively fast and comes with all the decisions that directly impact on the variables that are of interest to the movement scientist. Therefore, the user may want to run g.part1 overnight or on a computing cluster, while g.part2 can then be the main playing ground for the movement scientist. Function g.shell.GGIR provides the main shell that allows for operating g.part1 and g.part2.

Usage

```
g.part1(datadir=c(),outputdir=c(),f0=1,f1=c(),windowsizes = c(5,900,3600),
                    desiredtz = "Europe/London",chunksize=c(),studyname=c(),
                    do.enmo = TRUE,do.lfenmo = FALSE,do.en = FALSE,
                    do.bfen = FALSE,do.hfen=FALSE,do.hfenplus = FALSE,
                    do.teLindert2013=FALSE,do.anglex=FALSE,do.angley=FALSE,
                    do.anglez=FALSE,do.enmoa=FALSE,
                    do.roll_med_acc_x=FALSE,do.roll_med_acc_y=FALSE,
                    do.roll_med_acc_z=FALSE,do.dev_roll_med_acc_x=FALSE,
                    do.dev_roll_med_acc_y=FALSE,do.dev_roll_med_acc_z=FALSE,
                    do.cal = TRUE,lb = 0.2, hb = 15, n = 4,
                    use.temp=TRUE,spherecrit=0.3,minloadcrit=72,
                    printsummary=TRUE,print.filename=FALSE,overwrite=FALSE,
                    backup.cal.coef=c(),selectdaysfile=c(),dayborder=0)
```

Arguments

`datadir`	Directory where the accelerometer files are stored or list of accelerometer filenames and directories
`outputdir`	Directory where the output needs to be stored. Note that this function will attempt to create folders in this directory and uses those folder to organise output
`f0`	File index to start with (default = 1). Index refers to the filenames sorted in increasing order
`f1`	File index to finish with (defaults to number of files available)
`windowsizes`	see g.getmeta
`desiredtz`	see g.getmeta

chunksize — see g.getmeta

studyname — If the datadir is a folder then the study will be given the name of the data directory. If datadir is a list of filenames then the studyname will be used as name for the analysis

do.bfen — if TRUE, calculate metric BFEN with band-pass filter configuration set by lb and hb, see g.getmeta

do.enmo — if TRUE (default), calculate metric ENMO, see g.getmeta

do.lfenmo — if TRUE, calculate metric LFENMO with low-pass filter configuration set by hb,see g.getmeta

do.en — if TRUE, calculate metric EN, see g.getmeta

do.hfen — if TRUE, calculate metric HFEN with low-pass filter configuration set by hb, see g.getmeta

do.hfenplus — if TRUE, calculate metric HFENplus with band-pass filter configuration set by lb and hb, see g.getmeta

do.teLindert2013 — if TRUE, calculate the 5 sec epoch Actiwatch count replication as described by te Lindert et al 2013 in the journal SLEEP (volume 36, issue 5, page 781)

do.anglex — if TRUE, calculate the angle of the x-axis relative to the horizontal plane (degrees) utilizing all three axes

do.angley — if TRUE, calculate the angle of the y-axis relative to the horizontal plane (degrees) utilizing all three axes

do.anglez — if TRUE, calculate the angle of the z-axis relative to the horizontal plane (degrees) utilizing all three axes

do.enmoa — if TRUE (default), calculate metric ENMOa which is equal to metric ENMO but with the absolute taken from the Euclidean norm minus one.

do.roll_med_acc_x — see g.getmeta

do.roll_med_acc_y — see g.getmeta

do.roll_med_acc_z — see g.getmeta

do.dev_roll_med_acc_x — see g.getmeta

do.dev_roll_med_acc_y — see g.getmeta

do.dev_roll_med_acc_z — see g.getmeta

do.cal — Whether to apply auto-calibration or not, see g.calibrate. Default and recommended setting is TRUE

lb — lower boundary of the frequency filter (in Hertz)

hb — upper boundary of the frequency filter (in Hertz), see g.getmeta

n — order of the frequency filter, see g.getmeta

`use.temp` see g.calibrate use temperature sensor data if available (Geneactive only)

`spherecrit` see g.calibrate the minimum required acceleration value (in g) on both sides of 0 g for each axis. Used to judge whether the sphere is sufficiently populated

`minloadcrit` see g.calibrate the minimum number of hours the code needs to read for the autocalibration procedure to be effective (only sensitive to multitudes of 12 hrs, other values will be ceiled). After loading these hours only extra data is loaded if calibration error has not be reduced to under 0.01 g.

`printsummary` see g.calibrate if TRUE will print a summary when done

`print.filename` Whether to print the filename before before analysing it (default is FALSE). Printing the filename can be useful to investigate problems (e.g. to verify that which file is being read).

`overwrite` Overwrite previously generated milestone data by this function for this particular dataset. If FALSE then it will skip the previously processed files (default = FALSE).

`backup.cal.coef`
If the auto-calibration fails then the user has the option to provide back-up calibration coefficients via this argument. The value of the argument needs to be the name and directory of a csv-spreadsheet with the following column names and subsequent values: 'filename' with the names of accelerometer files on which the calibration coefficients need to be applied in case auto-calibration fails; 'scale.x', 'scale.y', and 'scale.z' with the scaling coefficients; 'offset.x', 'offset.y', and 'offset.z' with the offset coefficients, and; 'temperature.offset.x', 'temperature.offset.y', and 'temperature.offset.z' with the temperature offset coefficients. The argument is intended for analysing short lasting laboratory experiments with insufficient sphere data, but for which calibration coefficients can be derived in an alternative way. It is the users responsibility to compile the csv-spreadsheet.

`selectdaysfile` Optional functionality. Character pointing at a csv file holding the relationship between device serial numbers (first column) and measurement dates of interest (second and third column). The date format should be dd/mm/yyyy. And the first row if the csv file is assumed to have a character variable names, e.g. "serialnumber" "Day1" and "Day2" respectively. Raw data will be extracted and stored in the output directory in a new subfolder named 'raw'.

`dayborder` Hour at which days start and end (default = 0), value = 4 would mean 4am

Value

The function provides no values, it only ensures that the output from other functions is stored in .RData(one file per accelerometer file) in folder structure

Author(s)

Vincent T van Hees <vincentvanhees@gmail.com>

References

- van Hees VT, Gorzelniak L, Dean Leon EC, Eder M, Pias M, et al. (2013) Separating Movement and Gravity Components in an Acceleration Signal and Implications for the Assessment of Human Daily Physical Activity. PLoS ONE 8(4): e61691. doi:10.1371/journal.pone.0061691
- van Hees VT, Fang Z, Langford J, Assah F, Mohammad A, da Silva IC, Trenell MI, White T, Wareham NJ, Brage S. Auto-calibration of accelerometer data for free-living physical activity assessment using local gravity and temperature: an evaluation on four continents. J Appl Physiol (1985). 2014 Aug 7

Examples

```
## Not run:
datafile = "C:/myfolder/mydata"
outputdir = "C:/myresults"
g.part1(datadir,outputdir)

## End(Not run)
```

`g.part2` *function to analyse and summarize pre-processed output from g.part1*

Description

Loads the output from g.part1 and then applies g.impute and g.analyse, after which the output is converted to .RData-format which will be used by g.shell.GGIR to generate reports. The variables in these reports are the same variables as described in g.analyse.

Usage

```
g.part2(datadir=c(),metadatadir=c(),f0=c(),f1=c(),strategy = 1, hrs.del.start = 0.5,
hrs.del.end = 0.5, maxdur = 7, includedaycrit = 16,
L5M5window = c(0,24), M5L5res = 10, winhr = 5, qwindow=c(0,24), qlevels = c(0.1),
ilevels = c(0,10), mvpathreshold = c(100),boutcriter = 0.8,ndayswindow=7,idloc=1,
do.imp=TRUE,storefolderstructure=FALSE,overwrite=FALSE,epochvalues2csv=FALSE,
mvpadur=c(1,5,10),selectdaysfile=c(),window.summary.size=10,dayborder=0,
mvpa.2014 = FALSE,closedbout=FALSE,desiredtz="Europe/London")
```

Arguments

`datadir`	Directory where the accelerometer files are stored or list, e.g. "C:/mydata" of accelerometer filenames and directories, e.g. c("C:/mydata/myfile1.bin","C:/mydata/myfile2.bin").
`metadatadir`	Directory where the output from g.part1 was stored
`f0`	File index to start with (default = 1). Index refers to the filenames sorted in increasing order
`f1`	File index to finish with (defaults to number of files available)

strategy	how to deal with knowledge about study protocol. value = 1 to select data based on hrs.del.start, hrs.del.end, and maxdur. Value = 2 to only use the data between the first midnight and the last midnight, value = 3 only selects the most active X days in the files. X is specified by argument ndayswindow See also g.impute
hrs.del.start	how many HOURS after start of experiment did wearing of monitor start?, see g.impute
hrs.del.end	how many HOURS before the end of the experiment did wearing of monitor definitely end?, see g.impute
maxdur	how many DAYS after start of experiment did experiment definitely stop? (set to zero if unknown = default), see g.impute
includedaycrit	minimum required number of valid hours in day specific analysis (NOTE: there is no minimum required number of hours per day in the summary of an entire measurement, every available hour is used to make the best possible inference on average metric value per week)
L5M5window	start and end time, in 24 hour clock hours, over which L5M5 needs to be calculated. The calculation is done based on the average day
M5L5res	resoltion of L5 and M5 analysis in minutes (default: 10 minutes)
winhr	window size in hours of L5 and M5 analysis (dedault = 5 hours)
qwindow	start and end time, in 24 hour clock hours, over which distribution in metric values need to be extracted. Value = c(0,24) will consider all 24 hours.
qlevels	array of percentiles for which value needs to be extracted. These need to be expressed as a fraction of 1, e.g. c(0.1, 0.5, 0.75). There is no limit to the number of percentiles. If left empty then percentiles will not be extracted. Distribution will be derived from short epoch metric data, see g.getmeta.
ilevels	Levels for acceleration value frequency distribution in mg, e.g. c(0,100,200) There is no constriction to the number of levels.
mvpathreshold	Threshold for MVPA estimation. Threshold needs to be based on metric ENMO. This can be a single number or an array of numbers, e.g. c(100,120). In the later case the code will estimate MVPA seperately for each threshold. If this variable is left blank c() then MVPA is not estimated
boutcriter	The variable boutcriter is a number between 0 and 1 and defines what fraction of a bout needs to be above the mvpathreshold
ndayswindow	If strategy is set to 3 then this is the size of the window as a number of days
idloc	If value = 1 (default) the code assumes that ID number is stored in the obvious header field. If value = 2 the code uses the character string preceding the character '_' in the filename as the ID number
do.imp	Whether to impute missing values (e.g. suspected of monitor non-wear) or not by g.impute. Default and recommended setting is TRUE
storefolderstructure	Store folder structure of the accelerometer data
overwrite	Overwrite previously generated milestone data by this function for this particular dataset. If FALSE then it will skip the previously processed files (default = FALSE).

epochvalues2csv
: If TRUE then epoch values are exported to a CSV spreadsheet. Here, non-wear time is imputed where possible (default = FALSE).

mvpadur
: default = c(1,5,10). Three bout duration for which MVPA will be calculated

selectdaysfile
: Functionality designed for the London Centre of Longidutinal studies. Csv file holding the relation between device serial numbers and measurement days of interest.

dayborder
: Hour at which days start and end (default = 0), value = 4 would mean 4am

window.summary.size
: Functionality designed for the London Centre of Longidutinal studies. Size in minutes of the summary window

mvpa.2014
: If TRUE use the MVPA bout definition as has been available since 2014. To use the new calculation set value to FALSE (default) and also set argument closedbout. The 2014 definition looked at percentage of time (boutcritir) above a threshold (mvpathreshold) per standardised time duration (values of mvpadur). The newer bout defintion ooks at the percentage of time above a threshold for the entire bout regardless of its duration. Literature provides little guidance on which implementation is better. This could be an area of future research, but for the moment I implemented both approaches. I do not think either of them is wrong, they just require a different interpretation and are incompatible.

closedbout
: If TRUE then count breaks in a bout towards the bout duration. If FALSE then only count time spent above the threshold towards the bout duration.

desiredtz
: see g.getmeta

Value

The function provides no values, it only ensures that other functions are called and that their output is stored in the folder structure as created with g.part1.

Author(s)

Vincent T van Hees <vincentvanhees@gmail.com>

References

- van Hees VT, Gorzelniak L, Dean Leon EC, Eder M, Pias M, et al. (2013) Separating Movement and Gravity Components in an Acceleration Signal and Implications for the Assessment of Human Daily Physical Activity. PLoS ONE 8(4): e61691. doi:10.1371/journal.pone.0061691
- van Hees VT, Fang Z, Langford J, Assah F, Mohammad A, da Silva IC, Trenell MI, White T, Wareham NJ, Brage S. Auto-calibration of accelerometer data for free-living physical activity assessment using local gravity and temperature: an evaluation on four continents. J Appl Physiol (1985). 2014 Aug 7

Examples

```
## Not run:
metadatadir = "C:/myresults/output_mystudy"
```

```
g.part2(metadatadir)

## End(Not run)
```

`g.part3` *Detection of sustained inactivity periods as needed for sleep detection in g.part4.*

Description

Function called by g.shell.GGIR. It estimates the sustained inactivity periods in each day, which are used as input for g.part4 which then labels them as nocturnal sleep or day time sustained inactivity periods. Typical users should work with function g.shell.GGIR only.

Usage

```
g.part3(metadatadir=c(),f0,f1,anglethreshold = 5,timethreshold = 5,
  ignorenonwear=FALSE,overwrite=FALSE,desiredtz="Europe/London")
```

Arguments

`metadatadir`	Directory that holds a folder 'meta' and inside this a folder 'basic' which contains the milestone data produced by g.part1. The folderstructure is normally created by g.part1 and g.shell.GGIR will recognise what the value of metadatadir is.
`f0`	File index to start with (default = 1). Index refers to the filenames sorted in increasing order
`f1`	File index to finish with (defaults to number of files available)
`anglethreshold`	Angle threshold (degrees) for sustained inactivity periods detection, default = 5
`timethreshold`	Time threshold (minutes) for sustained inactivity periods detection, default = 5. This can be specified as multiple thresholds, each of which will be implemented. For example, timethreshold = c(5,10)
`ignorenonwear`	If TRUEE then ignore detected monitor non-wear periods to avoid confusion between monitor non-wear time and sustained inactivity (default = TRUE)
`overwrite`	Overwrite previously generated milestone data by this function for this particular dataset? If FALSE then it will skip the previously processed files (default = FALSE).
`desiredtz`	See g.getmeta

Value

The function provides no values, it only ensures that other functions are called and that their output is stored in .RData files.

- `night` nightnumber

- definition definition of sustained inactivity. For example, T10A5 refers to 10 minute window and a 5 degree angle (see paper for further explaination).
- start.time.day timestamp when the day started
- nsib.periods number of sustained inactivity bouts
- tot.sib.dur.hrs total duration of all sustained inactivity bouts
- fraction.night.invalid fraction of the night for which accelerometer data was invalid, e.g. monitor not worn
- sib.period number of sustained inactivity period
- sib.onset.time onset time of sustained inactivity period
- sib.end.time end time of sustained inactivity period

Author(s)

Vincent T van Hees <vincentvanhees@gmail.com>

References

- van Hees VT, Sabia S, et al. (2015) A novel, open access method to assess sleep duration using a wrist-worn accelerometer, PLoS ONE, November 2015

Examples

```
## Not run:
metadatadir = "C:/myfolder/meta" # assumes that there is a subfolder in
# metadatadir named 'basic' containing the output from g.part1
g.part3(metadatadir=metadatadir,anglethreshold=5,timethreshold=5,overwrite=FALSE)

## End(Not run)
```

---

g.part4 *Labels detected sustained inactivity periods by g.part3 as either nocturnal sleep or daytime sustained inactivity*

Description

Loads output from g.part3 as stored in milestone data and sleep log information (if available) and then uses these information sources to define nocturnal sleep and daytime sustained inactivity.

Usage

```
g.part4(datadir=c(),metadatadir=c(),f0=f0,f1=f1,idloc=1,
loglocation = c(),colid = 1,coln1 = 9,nnights = 7,
sleeplogidnum=FALSE,do.visual=FALSE,outliers.only = FALSE,
    excludefirstlast=FALSE,criterror = 1,includenightcrit=4,
    relyonsleeplog=FALSE,def.noc.sleep=c(),
    storefolderstructure=FALSE,overwrite=FALSE)
```

Arguments

datadir	Directory where the accelerometer files are stored or list of accelerometer filenames and directories
metadatadir	Directory that holds a folders 'meta' and inside this a folder 'basic' which contains the milestone data produced by g.part1. The folderstructure is normally created by g.part3. When using g.part4 via g.shell.GGIR then g.shell.GGIR will automatically recognise what the value of metadatadir is, so the user does not need to specify this.
f0	File index to start with (default = 1). Index refers to the filenames sorted in increasing order
f1	File index to finish with (defaults to number of files available)
idloc	If value = 1 (default) the code assumes that ID number is stored in the obvious header field. If value = 2 the code uses the character string preceding the character '_' in the filename as the ID number
loglocation	Location of the spreadsheet (csv) with sleep log information. The spreadsheet needs to have the following structure: one column for participant id, and then followed by alternatingly one column for onset time and one column for waking time. There can be multiple sleeplogs in the same spreadsheet. The first raw of the spreadsheet needs to be filled with column names, it does not matter what these column names are. Timestamps are to be stored without date as in hh:mm:ss. If onset corresponds to lights out or intention to fall asleep, then it is the end-users responsibility to account for this in the interpretation of the results.
colid	Column number in the sleep log spreadsheet in which the participant ID code is stored (default = 1)
coln1	Column number in the sleep log spreadsheet where the onset of the first night starts
nnights	Number of nights for which sleep log information should be available. It assumes that this is constant within a study. If sleep log information is missing for certain nights then leave these blank
sleeplogidnum	Should the participant identifier as stored in the sleeplog be interpretted as a number (TRUE=default) or a character (FALSE)?
do.visual	If g.part4 is run with do.visual == TRUE then the function will generate a pdf with a visual representation of the overlap between the sleeplog entries and the accelerometer detections. This can be used to visualy verify that the sleeplog entries do not come with obvious mistakes.
outliers.only	Relevant for do.visual == TRUE. Outliers.only == FALSE will visualise all available nights in the data. Outliers.only == TRUE will visualise only for nights with a difference in onset or waking time larger than the variable of argument criterror.
excludefirstlast	If TRUE then the first and last night of the measurement are ignored for the sleep assessment.
criterror	Relevant for do.visual == TRUE and outliers.only == TRUE. criterror specifies the number of minimum number of hours difference between sleep log and accelerometer estimate for the night to be included in the visualisation

includenightcrit
: Minimum number of valid hours per night (24 hour window between noon and noon)

relyonsleeplog
: If TRUE then sleep onset and waking time are defined based on timestamps derived from sleep log if FALSE (default) the sleep log is only used to guide the accelerometer-based detection. If participants were instructed NOT to wear the accelerometer during waking hours then set to TRUE, in all other scenarios set to FALSE (FALSE).

def.noc.sleep
: The time window during which sustained inactivity will be assumed to represent sleep, e.g. def.noc.sleep=c(21,9). This is only used if no sleep log entry is available. If def.noc.sleep is left blank then the 12 hour window centred at the least active 5 hours of the 24 hour period will be used instead.

storefolderstructure
: Store folder structure of the accelerometer data

overwrite
: Overwrite previously generated milestone data by this function for this particular dataset. If FALSE then it will skip the previously processed files (default = FALSE).

Value

The function does not produce values but generates an RData file in the milestone subfolder ms4.out which incudes a dataframe named `nightsummary` and comes with the following variables:

- `id` Participant id extracted from file
- `night` Night number
- `acc_onset` Detected onset of sleep expressed as hours since the previous midnight
- `acc_wake` Detected waking time (after sleep period) expressed as hours since the previous midnight
- `acc_timeinbed` Difference between onset and waking time.
- `acc_def` Definition of sustained inactivity by accelerometer
- `sleeplog_onset` Sleep onset derived from sleep log or specified by researcher if not sleep log is available
- `sleeplog_wake` Waking time derived from sleep log or specified by researcher if not sleep log is available
- `sleeplog_timeinbed` Time in bed derived from sleep log or specified by researcher if not sleep log is available
- `error_onset` Difference between sleep onset as estimated by accelerometer and estimated by sleeplog/defined by researcher
- `error_wake` Difference between waking time as estimated by accelerometer and estimated by sleeplog/defined by researcher
- `fraction_night_invalid` Fraction of the night for which the data was invalid, e.g. monitor not worn or no accelerometer measurement started/ended within the night

- acc_dur_noc Total sleep duration, which equals the accumulated nocturnal sustained inactivity bouts. This is not the same as the time in bed, which only looks at time difference between falling asleep and waking up.
- acc_dur_sibd Accumulated sustained inactivty bouts during the day. These are the periods we would label during the night as sleep, but during the day they form a subclass of inactivity, which may represent day time sleep or wakefulness while being motionless for a sustained period of time
- acc_n_noc Number of noturnal sleep periods
- acc_n_sibd Number of sustained inactivity periods during the day
- acc_onset_ts acc_onset formatted as a timestamp
- acc_wake_ts acc_wake formatted as a timestamp
- sleeplog_onset_ts sleeplog_onset formatted as a timestamp
- sleeplog_wake_ts sleeplog_wake formatted as a timestamp
- page pdf page on which the visualisation can be found
- daysleeper If 0 then the person is a nightsleeper (sleep period did not overlap with noon) if value=1 then the person is a daysleeper (sleep period did overlap with noon)
- weekday Day of the week on which the night started
- calendardate Calendar data which the night started
- filename Name of the accelerometer file
- cleaningcode 0: no problem; 1: sleeplog not available, 2: not enough valid accelerometer data, 3: no accelerometer data available, 4: there were no nights to be analysed for this person
- sleeplog_used Whether a sleep log was used (TRUE/FALSE)
- acc_available Whether accelerometer data was available (TRUE/FALSE). This dataframe is used in g.report.part4 to create two reports one per night and one per person.

Note that function g.shell.GGIR comes with the option for report generation. In relation to function g.part4 it is important to mention that these reports are effectively the variable names mentioned above or derivatives. Please fine below extra clarification on a few of the variable names for which the meaning may not be obvious:

- sleeplog_used Whether a sleeplog was available (TRUE) or not (FALSE)
- n_nights_acc Number of nights of accelerometer data
- n_WE_nights_complete Number of weekend nights complete which means both accelerometer and sleeplog data
- n_WD_nights_complete Number of weekday nights complete which means both accelerometer and sleeplog data
- n_WEnights_daysleeper Number of weekend nights on which the person slept until after noon
- n_WDnights_daysleeper Number of weekday nights on which the person slept until after noon
- sleeplog_dur_AD_mn Mean sleep duration according to sleeplog accros all days

- `sleeplog_dur_AD_sd` Standard deviation of sleep duration according to sleeplog accros all days
- `sleeplog_dur_WD_sd` Standard deviation of sleep duration according to sleeplog accros week-days
- `sleeplog_dur_WE_sd` Standard deviation of sleep duration according to sleeplog accros week-end days

Author(s)

Vincent T van Hees <vincentvanhees@gmail.com>

References

- van Hees VT, Sabia S, et al. (2015) A novel, open access method to assess sleep duration using a wrist-worn accelerometer, PLoS ONE, accepted for publication October 2015

Examples

```
## Not run:
metadatadir = "C:/myfolder/meta" # assumes that there is a subfolder in
# metadatadir named 'ms3.out' containing the output from g.part3
g.part4(metadatadir=metadatadir)

## End(Not run)
```

---

g.plot *function to generate a plot for quality check purposes*

Description

Function takes meta-data as generated by g.getmeta and g.impute to create a visual representation of imputed time periods

Usage

```
g.plot(IMP, M, I, durplot)
```

Arguments

`IMP`	output from g.impute
`M`	output from g.getmeta
`I`	output from g.inspectfile
`durplot`	number of days to plot

Value

function only produces a plot, no values

Author(s)

Vincent T van Hees <vincentvanhees@gmail.com>

Examples

```
## Not run:
  #inspect file:
  I = g.inspectfile(datafile)

  #autocalibration:
  C = g.calibrate(datafile)

  #get meta-data:
  M = g.getmeta(datafile)

## End(Not run)
data(data.getmeta)
data(data.inspectfile)

#impute meta-data:
IMP = g.impute(M = data.getmeta, I = data.inspectfile, strategy = 1,
hrs.del.start = 0, hrs.del.end = 0, maxdur = 0)

#plot data
g.plot(IMP, M = data.getmeta, I = data.inspectfile, durplot=4)
```

---

g.shell.GGIR *Shell function for analysing a accelerometer dataset.*

Description

This function is designed to help users operate all steps of the analysis. It helps to generate and structure milestone data, produces user-friendly reports. The function acts as a shell with calls to g.part1, g.part2, g.part3 and g.part4. Please see these specific functions for clarification on optional input arguments.

Usage

```
g.shell.GGIR(mode=c(1,2),datadir=c(),outputdir=c(),studyname=c(),f0=1,f1=0,
do.report=c(2),overwrite=FALSE,visualreport=FALSE,viewingwindow=1,...)
```

Arguments

mode	Specify which of the four parts need to be run, e.g. mode = 1 makes that g.part1 is run. Default setting, mode = c(1,2), makes that both part1 and part2 are ran. Note that if mode = c(1,3) then the code will also set do.anglez = TRUE in order to enable sleep detection. If you run part 1 and 3 seperatedly then you need to think about setting anglez to TRUE when running part1.

datadir	Directory where the accelerometer files are stored or list, e.g. "C:/mydata" of accelerometer filenames and directories, e.g. c("C:/mydata/myfile1.bin","C:/mydata/myfile2.bin").
outputdir	Directory where the output needs to be stored. Note that this function will attempt to create folders in this directory and uses those folder to keep output
studyname	If the datadir is a folder then the study will be given the name of the data directory. If datadir is a list of filenames then the studyname as specified by this input argument will be used as name for the study
f0	File index to start with (default = 1). Index refers to the filenames sorted in increasing order
f1	File index to finish with (defaults to number of files available)
overwrite	Do you want to overwrite analysis for which milestone data exists? If overwrite=FALSE then milestone data from a previous analysis will be used if available and visual reports will not be created again.
do.report	For which parts to generate a summary spreadsheet: 2 and/or 4. Default is c(2). A report will be generated based on the available milestone data. When creating milestone data with multiple machines it is advisable to turn the report generation off when generating the milestone data, value = c(), and then to merge the milestone data and turn report generation back on while setting overwrite to FALSE.
visualreport	If TRUE then generate visual report based on combined output from part 2 and 4. This is in beta-version at the moment.
viewingwindow	Centre the day as displayed around noon (value = 1) or around midnight (value = 2)
...	Any input argument needed for functions g.part1, g.part2, g.part3 or g.part4. See respective function documentation for further clarification.

Value

The function provides no values, it only ensures that other functions are called and that their output is stored.

Author(s)

Vincent T van Hees <vincentvanhees@gmail.com>

References

- van Hees VT, Gorzelniak L, Dean Leon EC, Eder M, Pias M, et al. (2013) Separating Movement and Gravity Components in an Acceleration Signal and Implications for the Assessment of Human Daily Physical Activity. PLoS ONE 8(4): e61691. doi:10.1371/journal.pone.0061691
- van Hees VT, Fang Z, Langford J, Assah F, Mohammad A, da Silva IC, Trenell MI, White T, Wareham NJ, Brage S. Auto-calibration of accelerometer data for free-living physical activity assessment using local gravity and temperature: an evaluation on four continents. J Appl Physiol (1985). 2014 Aug 7
- van Hees VT, Sabia S, et al. (2015) A novel, open access method to assess sleep duration using a wrist-worn accelerometer, PLoS ONE, November 2015

Examples

```
## Not run:
mode= c(1,2,3,4)
datadir= "C:/myfolder/mydata"
outputdir= "C:/myresults"
studyname="test"
f0 = 1
f1 = 2
g.shell.GGIR(#------------------------------
             # General parameters
             #------------------------------
             mode=mode,
             datadir=datadir,
             outputdir=outputdir,
             studyname=studyname,
             f0=f0,
             f1=f1,
             overwrite = FALSE,
             do.imp=TRUE,
             idloc=1,
             print.filename=FALSE,
             storefolderstructure = FALSE,
             #------------------------------
             # Part 1 parameters:
             #------------------------------
             windowsizes = c(5,900,3600),
             do.cal=TRUE,
             do.enmo = TRUE,
             do.anglez=TRUE,
             chunksize=1,
             printsummary=TRUE,
             #------------------------------
             # Part 2 parameters:
             #------------------------------
             strategy = 1,
             ndayswindow=7,
             hrs.del.start = 1,
             hrs.del.end = 1,
             maxdur = 9,
             includedaycrit = 16,
             L5M5window = c(0,24),
             M5L5res = 10,
             winhr = c(5,10),
             qlevels = c(c(1380/1440),c(1410/1440)),
             qwindow=c(0,24),
             ilevels = c(seq(0,400,by=50),8000),
             mvpathreshold =c(100,120),
             #------------------------------
             # Part 3 parameters:
             #------------------------------
             timethreshold= c(5,10),
             anglethreshold=5,
```

```
                ignorenonwear = TRUE,
                #------------------------------
                # Part 4 parameters:
                #------------------------------
                excludefirstlast = TRUE,
                includenightcrit = 16,
                def.noc.sleep = c(),
                loglocation= "D:/sleeplog.csv",
                outliers.only = FALSE,
                criterror = 4,
                relyonsleeplog = FALSE,
                sleeplogidnum = TRUE,
                colid=1,
                coln1=2,
                do.visual = TRUE,
                nnights = 9,
                #-----------------------------------
                # Report generation
                #------------------------------
                do.report=c(2,4))

## End(Not run)
```

Index

∗Topic **datasets**
data.calibrate, 4
data.getmeta, 5
data.inspectfile, 5

data.calibrate, 4
data.getmeta, 5
data.inspectfile, 5

g.analyse, *2*, 6, *21*
g.binread, *3*, 10
g.calibrate, *2–4*, *6*, 11, *18–20*
g.getmeta, *2*, *5–7*, *12*, 13, *16*, *18, 19*, *22–24*, *29*
g.impute, *2*, *6*, 16, *21, 22*, *29*
g.inspectfile, *2*, *5, 6*, *16*, 17, *29*
g.part1, *2*, *12*, *15*, *18*, 18, *21*, *23*, *30*
g.part2, *2*, *18*, 21, *30*
g.part3, 24, *30*
g.part4, 25, *30*
g.plot, 29
g.shell.GGIR, *2*, *18*, *21*, 30
GGIR *(*GGIR-package*)*, 2
GGIR-package, 2