Package 'GGIR'

May 8, 2019

Type Package

Title Raw Accelerometer Data Analysis

Version 1.9-1

Date 2019-05-08

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

Description A tool to process and analyse data collected with wearable raw acceleration sensors as described in van Hees and colleagues (2014) <doi: 10.1152/japplphysiol.00421.2014> and (2015) <doi: 10.1371/journal.pone.0142533>. The package has been developed and tested for binary data from 'GENEActiv' <https://www.activinsights.com/> and GENEA devices (not for sale), .csv-export data from 'Actigraph' <http://actigraphcorp.com> devices, and .cwa and .wav-format data from 'Axivity' <https://axivity.com/product/ax3>. These devices are currently widely used in research on human daily physical activity.

URL https://github.com/wadpac/GGIR/,
https://groups.google.com/forum/#!forum/RpackageGGIR

BugReports https://github.com/wadpac/GGIR/issues

License LGPL (>= 2.0, < 3) | file LICENSE

Suggests MASS, signal, zoo, mmap, bitops, matlab, GENEAread, tuneR, testthat, covr, knitr, rmarkdown

Imports data.table, Rcpp (>= 0.12.10)

Depends stats, utils, R (>= 3.1.2)

NeedsCompilation yes

LinkingTo Rcpp

VignetteBuilder knitr

ByteCompile yes

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

Joan Capdevila Pujol [ctb],
Jairo H Migueles [ctb]

Repository CRAN

Date/Publication 2019-05-08 17:20:09 UTC

R topics documented:

`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.

For a tutorial and more background in formation on GGIR, please see the package vignette: Accelerometer data processing with GGIR

The package has been developed with and for the accelerometer brands Genea and GENEActive. Additionally, it should work for .csv data from GENEActiv, .csv data from Actigraph, and .wav, .csv () and .cwa format from AX3 (Axivity). 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.

A non-exhaustive overview of publications related to GGIR can be found here

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, ms4out, and ms5out for respectively g.part1, g.part2, g.part3, g.part4, and g.part5 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.

g.part5 finally takes the output from parts 2 and 4 to describe time spent between waking up in the morning and waking up the next day subdivided by behavioural category. g.part5 calculates for each of these categories the time spent, the number of bouts, the average acceleration and the number of blocks.

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:	1.9-1
Date:	2019-05-08
License:	LGPL (>= 2.0, < 3)
Discussion group:	https://groups.google.com/forum/#!forum/rpackageggir

Author(s)

- Vincent T van Hees <vincentvanhees@gmail.com> main developer
- Zhou Fang co-developed function g.calibrate
- Jing Hua Zhao <jinghua.zhao@mrc-epid.cam.ac.uk> co-developed function g.binread
- Joe Heywood helped develop the functionality to process only specific days
- Evgeny Mirkes developed function g.cwaread
- Severine Sabia tested and provided feedback on various functions
- Joan Capdevila Pujol helped to improve various function
- Jairo H Migueles <jairohm@ugr.es> helped to improve 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)
```

`chartime2iso8601` *Convert character timestamps to iso8601 timestamp*

Description

To avoid ambiguities when sharing and comparing timestamps. All timestamps are expressed in iso8601 format: https://en.wikipedia.org/wiki/ISO_8601

Usage

```
chartime2iso8601(x,tz)
```

Arguments

x	Vector of timestamps in character format: year-month-date and optional followed by hour:minute:second For example, "1980-01-01 18:00:00"
tz	Timezone of data collection, e.g. "Europe/London". See https://en.wikipedia.org/wiki/List_of_tz_databas for full list

Examples

```
x ="1980-1-1 18:00:00"
tz = "Europe/Amsterdam"
x_converted = chartime2iso8601(x,tz)
```

`create_test_acc_csv` *Creates csv data file for testing purposes*

Description

Creates file in the Actigraph csv data format with dummy data that can be used for testing. The file includes accelerometer data with bouts of higher acceleration, variations non-movement periods in a range of accelerometer positions to allow for testing the auto-calibration functionality.

Usage

```
create_test_acc_csv(sf=3,Nmin=2000,storagelocation=c())
```

Arguments

sf	Sample frequency in Hertz, the default here is low to minimize file size
Nmin	Number of minutes (minimum is 2000)
storagelocation	Location where the test file named testfile.csv will be stored If no value is provided then the function uses the current working directory

Value

The function does not produce any output values. Only the file is stored

Examples

```
## Not run:
  create_test_acc_csv()

## End(Not run)
```

---

create_test_sleeplog_csv

Creates csv sleeplog file for testing purposes

Description

Creates sleeplog file in the format as expected by g.part4 with dummy data (23:00 onset, 07:00 waking time for every night).

Usage

```
create_test_sleeplog_csv(Nnights=7,storagelocation=c())
```

Arguments

Nnights	Number of nights (minimum is 1)
storagelocation	Location where the test file named testfile.csv will be stored If no value is provided then the function uses the current working directory

Value

The function does not produce any output values. Only the file is stored

Examples

```
## Not run:
  create_test_sleeplog_csv()

## End(Not run)
```

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)
```

datadir2fnames *Generates vector of file names out of datadir input argument*

Description

Uses input argument datadir from g.part1 and the output from isfilelist to generate vector of filenames

Usage

```
datadir2fnames(datadir,filelist)
```

Arguments

datadir	See g.part1
filelist	Produced by isfilelist

Value

Character vector of filenames

Author(s)

Vincent T van Hees <vincentvanhees@gmail.com>

Examples

```
## Not run:
datadir2fnames(datadir = "C:/mydatafolder",filelist=TRUE)

## End(Not run)
```

`g.abr.day.names` *Abbreviates daynames to numbers, needed for report generation in g.plot5*

Description

Abbreviates daynames Monday becomes MON and Sunday becomes SUN

Usage

```
g.abr.day.names(daynames)
```

Arguments

`daynames`	Vector of daynames in character format

Author(s)

Vincent T van Hees <vincentvanhees@gmail.com>

Examples

```
daynames = c("Monday","Friday")
daynames_converted = g.abr.day.names(daynames)
```

`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,bout.metric = 1,
closedbout=FALSE,desiredtz = c(),
IVIS_windowsize_minutes = 60,
IVIS_epochsize_seconds = 3600, iglevels = 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	To specify windows over which all variables are calculated. If value = c(0,24) all variables will only be calculated over the full 24 hours in a day, If value =c(8,24) variables will be calculated over the window 0-8, 8-24 and 0-24. Previously this functionality was limited to the distribution in acceleration metric values, but now it also derives N valid hours, L5M5 analysis and MVPA.
quantiletype	type of quantile function to use (default recommended). For details, see quantile function in STATS package
L5M5window	Argument depricated after version 1.5-24. This argument used to define the start and end time, in 24 hour clock hours, over which L5M5 needs to be calculated. Now this is done with argument qwindow.
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
bout.metric	This argument used to be called mvpa.2014 and had TRUE or FALSE as its value. However, it has now become clear that this aspect of the analyses is still very much open for debate. Therefore, I have changed it into an argument where you can specify a metric for bout detection based on a number. A description of these bout metrics can be found in the new function g.getbout
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
IVIS_windowsize_minutes	Window size of the Intradaily Variability (IV) and Interdaily Stability (IS) metrics in minutes
IVIS_epochsize_seconds	Epoch size of the Intradaily Variability (IV) and Interdaily Stability (IS) metrics in seconds
iglevels	Levels for acceleration value frequency distribution in mg used for intensity gradient calculation (according to the method by Rowlands 2018). By default this is argument is empty and the intensity gradient calculation is not done. The user can either provide a single value (any) to make the intensity gradient use the bins c(seq(0,4000,by=25),8000) or the user could specify their own distribution. There is no constriction to the number of levels.

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_fullRecordingMean 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 (diurnal balanced), with invalid data imputed by the average at similar time points on different days of the week. In addition to ENMO it is possible to extract other acceleration metrics (i.e. BFEN, HFEN, HFENplus). We emphasize that it is calculated over the full recording because the alternative is that a variable is only calculated overmeasurement days with sufficient valid hours of data.
- pX_ENMO_mg_0-24h_fullRecording 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_fullRecording Starting time in hours of the least active five* hours within the time interval as specified (* window size defined by argument winhr)
- L5_ENMO_mg Average acceleration over L5
- M5hr_ENMO_mg_0-24_fullRecording Starting time in hours of the most active five* hours in the day within the time interval as specified (* window size defined by argument winhr) modifiable in g.getmeta)
- M5_ENMO_mg_0-24_fullRecording Average acceleration over M5
- ig_gradient_ENMO_0-24hr_fullRecording Intensity gradient calculated over the full recording.
- 1am-6am_ENMO_mg_fullRecording Average acceleration between 1am and 6am
- N valid WEdays Number of valid weekend days
- N valid WDdays Number of valid week days
- IS_interdailystability Intra daily variability

- IV_intradailyvariability Intra intradailyvariability
- 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
- ..._ENMO_mg... 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
- weekdayDay of the week
- measurementday Day number relative to start of the measurement
- 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
- 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
- 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

- 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
- mean_ENMO_mg_1-6am Mean acceleration between 1am and 6am
- mean_ENMO_mg_0-24hr 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_0-24hr 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
- ig_gradient_ENMO_0-24hr Gradient from intensity gradient analysis (Rowlands et al 2018) based on metric ENMO for the time segment 0 to 24 hours
- ig_intercept_ENMO_0-24hr Intercept from intensity gradient analysis (Rowlands et al 2018) based on metric ENMO for the time segment 0 to 24 hours
- ig_rsquared_ENMO_0-24hr r squared from intensity gradient analysis (Rowlands et al 2018) based on metric ENMO for the time segment 0 to 24 hours

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.applymetrics` *Extract metrics from acceleration signals*

Description

Function to extract metrics from acceleration signal. Not intended for direct use by user

Usage

```
g.applymetrics(Gx,Gy,Gz,n,sf,ws3,metrics2do)
```

Arguments

`Gx`	y acceleration signal
`Gy`	y acceleration signal
`Gz`	z acceleration signal
`n`	filter order, only needed if a metric is selected that involves a frequency filter
`sf`	sample frequency
`ws3`	Epoch size in seconds
`metrics2do`	Dataframe with Boolean indicator for all metrics whether they should be extracted or not. For instance, metrics2do$do.bfen = TRUE, indicates that the bfen metric should be extracted

Value

Dataframe with metric values in columns average per epoch (ws3)

Author(s)

Vincent T van Hees <vincentvanhees@gmail.com>

Examples

```
Gx = runif(n=10000,min=0,max=2)
Gy = runif(n=10000,min=1,max=3)
Gz = runif(n=10000,min=0,max=2)
metrics2do = data.frame(do.bfen=TRUE,do.enmo=TRUE,do.lfenmo=FALSE,
do.en=FALSE,do.hfen=FALSE,do.hfenplus=FALSE,do.mad=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)
extractedmetrics = g.applymetrics(Gx,Gy,Gz,n=4,sf=40,ws3=5,metrics2do)
```

`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, desiredtz = c())
```

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
`desiredtz`	see g.getmeta

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.create.sp.mat *Converts sleep period information. Not intended for direct use*

Description

Function to convert data into sleep period matrix part of g.part4.R. Not intended for direct use by package user

Usage

```
g.create.sp.mat(nsp,spo,sleepdet.t,daysleep=FALSE)
```

Arguments

nsp	Integer indicating the number of sleep periods
spo	Empty matrix with overview of sleep periods, 5 columns and as along as nps
sleepdet.t	Part of detected sleep from g.sib.det for one night and one sleep definition
daysleep	Boolean to indicator whether this person woke up after noon (daysleeper)

Value

- spo matrix with start and end of each sleep period
- calendardate date corresponding to the day on which the night started
- item wdayname weekdayname

Author(s)

Vincent T van Hees <vincentvanhees@gmail.com>

g.createcoordinates *Create coordinates for g.plot*

Description

Function creates the coordinates for the blocks g.plot Function not designed for direct use by package user.

Usage

```
g.createcoordinates(r,timeline)
```

Arguments

r	Vector of zeros and ones reflecting the moments in time when there should be a block (1)
timeline	Vector of time indicators, this can be numbers or actual timestamps The length of timeline needs to match the length of argument r

Value

List with two objects: x0 with all the coordinates correspoding to the start of each blocks on the timelines and x1 with all the coordinates corresponding to the end of each block on the timeline

Author(s)

Vincent van Hees <vincentvanhees@gmail.com>

---

g.cwaread *Function to read .cwa-format files as produced by the accelerometer named 'Axivity'*

Description

For reading .cwa-format data, if you have .wav format data then see function g.wavread

Usage

```
g.cwaread(fileName, start = 0, end = 0, progressBar = FALSE, desiredtz = c())
```

Arguments

fileName	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)
progressBar	Is trigger to switch on/off the text progress bar. If progressBar is TRUE then the function displays the progress bar but it works slightly slower
desiredtz	Desired timezone, see documentation g.getmeta

Value

data	dataframe with timestamp, raw x, -y, and, -z acceleration values, temperature, battery and light
header	file header

Author(s)

Evgeny Mirkes <em322@leicester.ac.uk>

g.detecmidnight *Detect all midnights in a time series*

Description

Detect all midnights in a time series

Usage

```
g.detecmidnight(time,desiredtz)
```

Arguments

time	Vector of timestamps, either in iso8601 or in POSIX format
desiredtz	See g.part2

Value

Output of the function is list containing the following objects:

- firstmidnight = timestamp of first midnight
- firstmidnighti = index of first midnight
- lastmidnight = timestamp of last midnight
- lastmidnighti = index of last midnight
- midnights = timestamps of midnights
- midnightsi = indeces of midnights

Author(s)

Vincent T van Hees <vincentvanhees@gmail.com>

g.dotorcomma *Assesses whether decimals in fileheader are stored with comma or dot separated decimals*

Description

The function is used by g.readaccfile to assess how numeric data should be interpretted

Usage

```
g.dotorcomma(inputfile,dformat,mon, desiredtz = c())
```

Arguments

inputfile	full path to inputfile
dformat	Data format code: 1=.bin, 2=.csv, 3=.wav, 4=.cwa
mon	Monitor code (accelorometer brand): 1=GENEA,2=GENEActiv,3=Actigraph, 4=Axivity.
desiredtz	Desired timezone, see documentation g.getmeta

Value

Character object showing how decimals are separated

Author(s)

Vincent T van Hees <vincentvanhees@gmail.com>

Examples

```
## Not run:
decn = g.dotorcomma(inputfile="C:/myfile.bin",dformat=1,mon=2)

## End(Not run)
```

---

g.downsample *Downsample a vector of numeric values at three time resolutions*

Description

Downsamples a vector of numeric values at three time resolutions: 1 seconds, ws3 seconds, and ws2 second. Function is not intended for direct interaction by package end user

Usage

```
g.downsample(sig,fs,ws3,ws2)
```

Arguments

sig	Vector of numeric values
fs	Sample frequency
ws3	ws3 epoch size, e.g. 5 seconds
ws2	ws2 epoch size, e.g. 90 seconds

Value

List with three object: var1, var2, and var3 corresponding to downsample time series at 1 seconds, ws2 seconds, and ws3 seconds resoluton, respectively

Author(s)

Vincent T van Hees <vincentvanhees@gmail.com>

Examples

```
 sig = runif(n=10000,min=1,max=10)
downsampled_sig = g.downsample(sig,fs=20,ws3=5,ws2=15)
```

---

g.extractheadervars *Extracts header variables from header object*

Description

Function is not intended for direct interaction by package end user

Usage

```
g.extractheadervars(I)
```

Arguments

I	Object produced by g.inspectfile

Value

- id = participant identifier
- iid = investigator identifier
- HN = handedness
- BL = body location
- SX = sex
- SN = serial number

Author(s)

Vincent T van Hees <vincentvanhees@gmail.com>

Examples

```
data(data.inspectfile)
headervars = g.extractheadervars(I=data.inspectfile)
```

g.getbout *function to calculate bouts from vector of binary classes*

Description

To detect bouts of behaviour in time series. The function is used by g.analyse

Usage

```
g.getbout(x, boutduration, boutcriter=0.8, closedbout=FALSE,
bout.metric=1,ws3=5)
```

Arguments

x	vector of zeros and/or ones to be screened for bouts of ones
boutduration	duration of bout in epochs
boutcriter	Minimum percentage of boutduration for which the epoch values are expected to meet the threshold criterium
closedbout	TRUE if you want breaks in bouts to be counted towards time spent in bouts (argument only active for bout.metric 1 and 2)
bout.metric	If value=1 the code uses the MVPA bout definition as has been available since 2014 (see papers by Sabia AJE 2014 and da Silva IJE 2014). Here, the algorithm looks for 10 minute windows in which more than XX percent of the epochs are above mvpathreshold, and then counts the entire window as mvpa. If value=2 the code 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 defition 1 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 definition 2 is: not counting breaks towards MVPA simplifies interpretation and still counts the two persons in the example as each others equal. If value=3, using sliding window across the data to test bout criteria per window and do not allow for breaks of 1 minute or longer. If value=4, same as 3 but also requires the first and last epoch to require the threshold criteria.
ws3	epoch length in seconds, only needed for bout.metric =3, because it needs to measure how many epochs equal 1 minute breaks

Value

x	Vector with binary numbers indicator where bouts where detected
boutcount	Vector with binary numbers indicator where bouts where detected and counted towards time spent in bouts, see argument closedbout for clarification

Author(s)

Vincent T van Hees <vincentvanhees@gmail.com>

Examples

```
y = g.getbout(x=round(runif(1000,0.4,1)),boutduration = 120,boutcriter=0.9,
    closedbout=FALSE,bout.metric=3,ws3=5)
```

---

g.getidfromheaderobject

Extracts participant identifier from header object

Description

Extracts participant identifier from header object, if it can not be found then the filename is used as identifier. Function is not intended for direct interaction by package end user

Usage

```
g.getidfromheaderobject(filename,header,dformat,mon)
```

Arguments

filename	File name
header	header object as extracted with g.inspectfile
dformat	Data format code, same as for g.dotorcomma
mon	Monitor code, same as for g.dotorcomma

Value

Participant identifier as character

Author(s)

Vincent T van Hees <vincentvanhees@gmail.com>

Examples

```
## Not run:
  g.getidfromheaderobject(filename="C:/myfile.bin",header,dformat=2,mon=2)

## End(Not run)
```

g.getM5L5 *Extract M5 and L5 from time series*

Description

Extract M5 and L5 from time series, function used by g.analyse and not intended for direct use by package user. Please see g.analyse for further clarification on functionalities

Usage

```
g.getM5L5(varnum,ws3,t0_LFMF,t1_LFMF,M5L5res,winhr)
```

Arguments

varnum	Numeric vector of epoch values
ws3	Small epoch size in seconds
t0_LFMF	Start hour of the day for the M5L5 analyses, e.g. 0 for midnight
t1_LFMF	End hour of the day for the M5L5 analyses, e.g. 24 for midnight
M5L5res	Resolution of hte M5L5 analyses in minutes
winhr	windowsize of M5L5 analyses, e.g. 5 hours

Value

- DAYL5HOUR = Starting time in hours of L5
- DAYL5VALUE = average acceleration during L5
- DAYM5HOUR = Starting time in hours of M5
- DAYM5VALUE = average acceleration during M5
- V5NIGHT = average acceleration between 1am and 6am

Author(s)

Vincent T van Hees <vincentvanhees@gmail.com>

Examples

```
data(data.getmeta)
g.getM5L5 = function(varnum=data.getmeta,ws3=5,t0_LFMF=0,
t1_LFMF=24,M5L5res=10,winhr=5)
```

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.mad = 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,dynrange=c(),...)
```

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.mad	if TRUE, calculate metric MAD (Mean Amplitude Deviarion)
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 4 Gb of RAM memory a value below 1 is recommended.

selectdaysfile	see g.part1
dayborder	see g.part1
dynrange	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
- Aittasalo M, Vaha-Ypya H, Vasankari T, Husu P, Jussila AM, and Sievanen H. Mean amplitude deviation calculated from raw acceleration data: a novel method for classifying the intensity of adolescents physical activity irrespective of accelerometer brand. BMC Sports Science, Medicine and Rehabilitation (2015).

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.getstarttime *Extract start time of a measurement*

Description

Extract start time of a measurement. GGIR calculates all timestamps by using the first timestamp and sample frequency. Not intended for direct use by package user

Usage

```
g.getstarttime(datafile,P,header,mon,dformat,desiredtz,selectdaysfile)
```

Arguments

datafile	Full path to data file
P	Object extracted with g.readaccfile
header	File header extracted with g.inspectfile
mon	Same as in g.dotorcomma
dformat	Same as in g.dotorcomma
desiredtz	Same as in g.dotorcomma
selectdaysfile	See g.part1

Value

The starttime

Author(s)

Vincent T van Hees <vincentvanhees@gmail.com>

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, desiredtz = c())
```

Arguments

`datafile`	name of data file
`desiredtz`	Desired timezone, see documentation g.getmeta

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.intensitygradient *Intensity gradient calculation*

Description

Calculates the intensity gradient based on Rowlands et al. 2018. The function assumes that the user has already calculated the value distribution.

Usage

```
g.intensitygradient(x,y)
```

Arguments

x	Numeric vector of mid-points of the bins (mg)
y	Numeric vector of time spent in bins (minutes)

Value

y_intercept	y-intercept of a linear regression line in log-log space
gradient	Beta coefficient of a linear regression line in log-log space
rsquared	R squared of x and y values in log-log space

Author(s)

Vincent T van Hees <vincentvanhees@gmail.com>

References

Rowlands A, Edwardson CL, et al. (2018) Beyond Cut Points: Accelerometer Metrics that Capture the Physical Activity Profile. MSSE 50(6):1. doi:10.1249/MSS.0000000000001561

g.loadlog *Load and clean sleeplog information*

Description

Loads sleeplog from a csv input file and applies sanity checks before storing the output in a dataframe

Usage

```
g.loadlog(loglocation=c(),coln1=c(),colid=c(),nnights=c(),
sleeplogidnum=TRUE)
```

Arguments

loglocation	See g.part4
coln1	See g.part4
colid	See g.part4
nnights	See g.part4
sleeplogidnum	See g.part4

Value

Data frame with sleeplog

Author(s)

Vincent T van Hees <vincentvanhees@gmail.com>

Examples

```
## Not run:
sleeplog = g.loadlog(loglocation="C:/mysleeplog.csv",coln1=2,
colid=1,nnights=5,sleeplogidnum=TRUE)

## End(Not run)
```

---

g.metric *Extract metrics from acceleration signals*

Description

Function to extract metrics from acceleration signal. Not intended for direct use by package user

Usage

```
g.metric(Gx,Gy,Gz,n=c(),sf,ii,TW=c(),lb=c(),hb=c(),gravity = 1)
```

Arguments

Gx	y acceleration signal
Gy	y acceleration signal
Gz	z acceleration signal
n	filter order, only needed if a metric is selected that involves a frequency filter
sf	sample frequency
ii	Integer to indicate which metric should be derived
TW	Time window size in samples used if the metric involves a time window
lb	Cut-off frequency corresponding to the lower boundary of frequency filter
hb	Cut-off frequency corresponding to the higher boundary of frequency filter
gravity	Size of gravity, default = 1

Value

Vector of metric values at the same time resolution as the input data

Author(s)

Vincent T van Hees <vincentvanhees@gmail.com>

Examples

```
Gx = runif(n=10000,min=0,max=2)
Gy = runif(n=10000,min=1,max=3)
Gz = runif(n=10000,min=0,max=2)
EuclideanNorm = g.metric(Gx,Gy,Gz,sf=40,ii=3,gravity = 1)
```

---

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.mad = 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,
        dynrange=c())
```

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.mad	if TRUE, calculate metric MAD (Mean Amplitude Deviation), see g.getmeta
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 4 am

dynrange
: Optional, provide dynamic range for accelerometer data to overwrite hardcoded 6 g for GENEA and 8 g for other brands

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
- Aittasalo M, Vaha-Ypya H, Vasankari T, Husu P, Jussila AM, and Sievanen H. Mean amplitude deviation calculated from raw acceleration data: a novel method for classifying the intensity of adolescents physical activity irrespective of accelerometer brand. BMC Sports Science, Medicine and Rehabilitation (2015).

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,
bout.metric=2,closedbout=FALSE,desiredtz="Europe/London",
IVIS_windowsize_minutes = 60, IVIS_epochsize_seconds = 3600,
iglevels = c())
```

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	Argument depricated after version 1.5-24. This argument used to define the start and end time, in 24 hour clock hours, over which L5M5 needs to be calculated. Now this is done with argument qwindow.
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	see g.analyse
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

`bout.metric` This argument used to be called mvpa.2014 and had TRUE or FALSE as its value. However, it has now become clear that this aspect of the analyses is still very much open for debate. Therefore, I have changed it into an argument where you can specify a metric for bout detection based on a number. A description of these bout metrics can be found in the new function g.getbout

`closedbout` See g.getbout

`desiredtz` see g.getmeta

`IVIS_windowsize_minutes` Window size of the Intradaily Variability (IV) and Interdaily Stability (IS) metrics in minutes

`IVIS_epochsize_seconds` Epoch size of the Intradaily Variability (IV) and Interdaily Stability (IS) metrics in seconds

`iglevels` see function g.analyse

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,
  acc.metric="ENMO", ignorenonwear=FALSE,overwrite=FALSE,
  desiredtz="Europe/London",constrain2range=TRUE,
  do.part3.pdf=TRUE)
```

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)
acc.metric	Which one of the metrics do you want to consider to analyze L5. The metric of interest need to be calculated in M (see g.part1)
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
constrain2range	Whether or not to constrain the range of threshold used in the diary free Sleep period time window detection
do.part3.pdf	Whether to generate a pdf for part 3 (default is TRUE). Turning this off could speed up the processing.

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
- van Hees VT, Sabia S, et al. (2018) Estimating sleep parameters using an accelerometer without sleep diary. Scientific Reports.

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=16,
    relyonsleeplog=FALSE,def.noc.sleep=c(),
    storefolderstructure=FALSE,overwrite=FALSE,
    desiredtz="Europe/London")
```

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 'def.noc.sleep=c()' then the 12 hour window centred at the least active 5 hours of the 24 hour period will be used instead. Here, L5 is hardcoded and will not change by changing argument winhr in function g.part2. If def.noc.sleep is filled with a single integer, e.g. def.noc.sleep=c(1) then the window will be detected with the method as described in van Hees et al. 2018 Scientific Reports.

`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).

`desiredtz` See g.getmeta

Details

The term sleeplog in variable names originates from the fact that the code was first developed in the presence of sleeplog data. Please interpret sleeplog as the method to derived the Sleep Period Time (SPT) window derived from (as applicable): sleeplog, HDCZA algorithm, L5R6 algorithm, or specified by researcher as a constant time interval.

There are, however, two exceptions: Variable sleeplog_used and n_nights_sleeplog truly refer to whether a sleep log was used and the number of nights on which a sleep log was used, respectively. I know this is confusing, but so far I have kept the variable names as they are to facilitate consistency in terminology.

If argument relyonsleeplog = FALSE, then the sleep parameter estimates from accelerometry, e.g. acc_onset, are guided by the sleeplog (or whichever method was used instead for SPT window estimation). In this example, the sleep onset time equals the start of the first sustained inactiviy bout that overlaps or follows the sleep onset time derived from the sleeplog (or whichever method was used instead for SPT window estimation).

If argument relyonsleeplog = TRUE, then the sleep onset estimates from accelerometry equals the estimate from the sleeplog.

Value

The function does not produce values but generates an RData file in the milestone subfolder ms4.out which incudes a dataframe named `nightsummary`. This dataframe is used in g.report.part4 to create two reports one per night and one per person. `nightsummary` 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 midnight of the previous night, see details.
- `acc_wake` Detected waking time (after sleep period) expressed as hours since the midnight of the previous night, see details.

- acc_SptDuration Difference between onset and waking time.
- acc_def Definition of sustained inactivity by accelerometer
- sleeplog_onset Start of Sleep Period Time window derived from (in order of priority) sleeplog, detected by the HDCZA algorithm, detected by L5HR6 algorithm, or specified by researcher,
- sleeplog_wake End of Sleep Period Time window derived from (in order of priority) sleeplog, detected by the HDCZA algorithm, detected by L5HR6 algorithm, or specified by researcher.
- sleeplog_SptDuration Time in bed derived from sleeplog_wake and sleeplog_onset.
- error_onset Difference between acc_onset and sleeplog_onset
- error_wake Difference between acc_wake and sleeplog_wake
- 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_SleepDurationInSpt Total sleep duration, which equals the accumulated nocturnal sustained inactivity bouts within the Sleep Period Time.
- 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, with nocturnal referring to the Sleep Period Time window.
- acc_n_sibd Number of sustained inactivity bouts during the day, with day referring to the time outside the Sleep Period Time window.
- 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 date on 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).

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)
- acc_eff Accelerometer detrive sleep efficiency within the sleep period time calculatd as the ratio between acc_SleepDurationInSpt and acc_SptDuration (denominator).
- n_nights_acc Number of nights of accelerometer data
- n_nights_sleeplog Number of nights of sleeplog data, see details
- 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
- acc_dur_msibd Average duration of the sustained inactivity bouts during the day (outside the sleep period duration). Calculated as acc_dur_sibd divided by acc_n_sibd per day, after which the mean and standard deviation are calculated across days.
- 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 weekdays
- sleeplog_dur_WE_sd Standard deviation of sleep duration according to sleeplog accros weekend days

Author(s)

Vincent T van Hees <vincentvanhees@gmail.com>

References

- van Hees VT, Sabia S, et al. (2018) AEstimating sleep parameters using an accelerometer without sleep diary, Scientific Reports.
- van Hees VT, Sabia S, et al. (2015) A novel, open access method to assess sleep duration using a wrist-worn accelerometer, PLoS ONE.

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.part5 *Merge output from physical activity and sleep analysis into one report*

Description

Function to merge the output from g.part2 and g.part4 into one report enhanced with profiling of sleep and physical activity stratified across intensity levels and based on bouted periods as well as non-bouted periods.

Usage

```
g.part5(datadir=c(), metadatadir=c(), f0=c(), f1=c(), strategy=1,
  maxdur=7, hrs.del.start=0, hrs.del.end =0,
  loglocation= c(), excludefirstlast.part5=FALSE,
  windowsizes=c(5,900,3600),acc.metric="ENMO", boutcriter.mvpa=0.8,
  boutcriter.in=0.9, boutcriter.lig=0.8, storefolderstructure=FALSE,
  threshold.lig = c(40), threshold.mod = c(100),
  threshold.vig = c(400), timewindow=c("MM","WW"),
  boutdur.mvpa = c(1,5,10), boutdur.in = c(10,20,30),
  boutdur.lig = c(1,5,10), winhr = 5, M5L5res = 10,
  overwrite=FALSE, desiredtz="Europe/London",
  bout.metric=4, dayborder=0, save_ms5rawlevels=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.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)
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, see g.impute
maxdur	how many DAYS after start of experiment did experiment definitely stop? (set to zero if unknown = default), see 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

`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. Timestamps are to be stored without date as in 18:20:00. 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.

`excludefirstlast.part5`
If TRUE then the first and last night of the measurement are ignored for the sleep assessment.

`windowsizes` see g.getmeta

`acc.metric` Which one of the metrics do you want to consider to describe behaviour. The metric of interest need to be calculated in M (see g.part1)

`boutcriter.mvpa`
A number between 0 and 1 and defines what fraction of a bout needs to be above the mvpathreshold

`boutcriter.in` A number between 0 and 1 and defines what fraction of a bout needs to be below the light threshold

`boutcriter.lig` A number between 0 and 1 and defines what fraction of a bout needs to be between the light and moderage threshold

`storefolderstructure`
Store folder structure of the accelerometer data

`threshold.lig` Threshold for light physical activity to separate inactivity from light. Value can be one number or an array of multiple numbers, e.g. threshold.lig =c(30,40). If multiple numbers are entered then analysis will be repliced for each combination of threshold values. Threshold is applied to the first metric in the milestone data, so if you have only specified do.ENMO == TRUE then it will be applied to ENMO.

`threshold.mod` Threshold for moderate physical activity to separate light from moderate. Value can be one number or an array of multiple numbers, e.g. threshold.mod =c(100,110). If multiple numbers are entered then analysis will be repliced for each ombination of threshold values. Threshold is applied to the first metric in the milestone data, so if you have only specified do.ENMO == TRUE then it will be applied to ENMO.

`threshold.vig` Threshold for vigorous physical activity to separate moderate from vigorous. Value can be one number or an array of multiple numbers, e.g. threshold.mod =c(400,500). If multiple numbers are entered then analysis will be repliced for each combination of threshold values. Threshold is applied to the first metric in the milestone data, so if you have only specified do.ENMO == TRUE then it will be applied to ENMO.

`timewindow` Timewindow over which summary statistics are derived. Value can be "MM" (midnight to midnight), "WW" (waking time to waking time), or both c("MM","WW").

`boutdur.mvpa` Durations of mvpa bouts in minutes to be extracted. The default values is c(1,5,10) and will start with the identification of 10 minute bouts, followed by 5 minute bouts in the rest of the data, and followed by 1 minute bouts in the rest of the data.

boutdur.in	Durations of inactivty bouts in minutes to be extracted. Inactivity bouts are detected in the segments of the data which were not labelled as sleep or MVPA bouts. The default duration values is c(10,20,30), this will start with the identification of 30 minute bouts, followed by 20 minute bouts in the rest of the data, and followed by 10 minute bouts in the rest of the data.
boutdur.lig	Durations of light activty bouts in minutes to be extracted. Light activity bouts are detected in the segments of the data which were not labelled as sleep, MVPA, or inactivity bouts. The default duration values is c(1,5,10), this will start with the identification of 10 minute bouts, followed by 5 minute bouts in the rest of the data, and followed by 1 minute bouts in the rest of the data.
M5L5res	resoltion of L5 and M5 analysis in minutes (default: 10 minutes)
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
bout.metric	See documnetion in g.getbout and
dayborder	Hour at which days start and end (default = 0), value = 4 would mean 4am
winhr	see g.getmeta
save_ms5rawlevels	boolean, whether to save the time series classification (levels) as a csv files

Details

This function writes all its output to an RData file The value `output` is a dataframe and comes with a large range of variables which hopefully are sufficiently intuitive or are already explained elsewhere in the package tutorial. When g.part5 is called from g.shell.GGIR with argument do.report = 5 then the output of g.part5 is conventiently stored in a csv spreadsheet. Therefore, you may not want/need to work with part5 directly.

Explaination of general terminology in the output of g.part5:

- acc_onset = onset of sleep according to accelerometer (+ diary) method expressed in hours since the midnight in the night preceding the night of interest, e.g. 26 is 2am.
- acc_wake = waking up time according to accelerometer (+ diary) method expressed in hours since the midnight in the night preceding the night of interest, e.g. 38 is 8am
- acc_onset_ts = onset of sleep according to accelerometer (+ diary) method expressed as a timestamp hours:minutes:seconds
- daysleep, if 0 then the person woke up before noon, if 1 then the person woke up after noon
- 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
- window_length_in_hours, this relates to the definition of a day either from waking up till waking up the next day or from midnight to midnight
- dur = duration

- acc = (average) acceleration
- nightwak = night waking
- SIB=sustained inactivity bouts, are the periods of time during which the accelerometer does not rotate at all for at least 5 or 10 minutes. This could be daytime sleep or the monitor not being worn for a very short period of time.
- OIN = other inactivity
- Nblock = number of blocks
- D10 = bout lengths with 10 minutes and longer
- T120 = threshold of 120 mg and higher
- _pla = plain average across available days
- _wei = weighted average across available days where weekend days always weighted 2/5 relative to the contribution of week days
- If there is no D in the variable name like in LIG50_120 then it refers to unbouted time spent between those thresholds
- dur_LIGB_D10T50_120 = the time spent in bouts of light activity of at least 10 minutes
- WW in filename refers to analyses based on the timewindow from waking to waking up
- MM in filename refers to analyses done on windows between midnight and midnight
- INB are bouts of time during which acceleration is below an acceleration threshold for at least X percent of the time.
- If boutdur.mvpa holds two bout duration thresholds like 1 and 10 minutes when it is set to c(1,10) then you will get D1T100 and D10T100. In this case D1T100 effectively means bouts between 1 and 9.99 minutes, while D10T100 refers to bouts of at least 10 minutes.
- TMOD and TVIG are total moderate and total vigorous PA with all bouted and unbouted time contributing. Just MOD and VIG are only the unbouted time spent in those categories.
- nightwak = night waking time
- Variable names with SIB in them are the sustained inactivity bouts that are not part of the inactivity bouts. Only TSIB shows the total time spent in all sustained inactivity bouts. I make this distinction, because sustained inactivity bouts, can be part of inactivity bouts and are then counted towards the inactivity bout (INB variables) and not towards the SIB category.
- TSIB = total of all sustained inactivity bouts (regardless of whether they contributed or not to inactivity bouts (INB).
- TOIN = Total other inactivity
- TIN = Total inactivity = TOIN + TSIB = Total other inactivty + Total sustained inactivity
- Levels of behaviour from least active to most active are: Sleep or SIB, Other inactivity, Light, Moderate, Vigorous
- Additionally they a grouped in: MVPA bouts (Moderate or Vigorous), Light bouts (just light), Inactivity bouts (Other inactivity or SIB), And the non-bouted time spent in each of the five categories, And also as the total of each category: TSIB, TOIN, TLIG, TMOD, TVIG
- dur_day_min_pla, dur_night_min_pla, and dur_nightandday_min_pla: duration of a day and night time

Example variable explanations: dur_day_OIN30_min is the time spent in minutes in other inactiivty during the day with a threshold of 30 mg not part of inactivity bouts, dur_day_MOD100_400_min is the time spent in moderate activity defined between 100 and 400mg but not part of an MVPA bout, dur_MVPA_D1T100_min is time spent in MVPA bouts defined as 100 mg or higher and lasting at least 1 minute and with no upper boundary if there is no other variable that starts with dur_MVPA_D, the existence of dur_INB_D10T30_min and dur_INB_D30T30_min indicates that dur_INB_D10T30_min corresponds to inactivity bouts lasting between 10 and 30 minutes and are defined by the threshold 30mg.

Motivation for default bout criteria for inactivity 0.9: Ssomewhat arbitrary decision, but the idea is that if you allow for bouts of 30 minutes it would not make sense to allow for breaks of 20 percent (6 minutes!) this is why I used a more stringent criteria for the highest category. Please note that you can change these criteria via arguments boutcriter.mvpa, boutcriter.in, and boutcriter.lig

Value

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

Author(s)

Vincent T van Hees <vincentvanhees@gmail.com>

References

- This function has not been described in a scientific journal yet

Examples

```
## Not run:
metadatadir = "C:/myfolder/meta"
g.part5(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.plot5 *Generate user-friendly visual report*

Description

Function called by g.shell.GGIR to generate report. Not intended for direct use by user

Usage

```
g.plot5(metadatadir=c(),dofirstpage=FALSE, viewingwindow = 1,
f0=c(),f1=c(),overwrite=FALSE,metric="ENMO",desiredtz = "Europe/London")
```

Arguments

metadatadir	See g.part2
dofirstpage	Boolean to indicate whether a first page with historgrams summarizing the whole measurement should be added
viewingwindow	See g.shell.GGIR
f0	See g.shell.GGIR
f1	See g.shell.GGIR
overwrite	See g.shell.GGIR
metric	Which one of the metrics do you want to consider to describe behaviour. The metric of interest need to be calculated in M (see g.part1)
desiredtz	See g.getmeta

Value

No values, this function only generates a plot

Author(s)

Vincent T van Hees <vincentvanhees@gmail.com>

Examples

```
  ## Not run:
  # generate plots for the first 10 files:
  g.plot5(metadatadir="C:/output_mystudy/meta/basic",dofirstpage=TRUE,
  viewingwindow = 1,f0=1,f1=10,overwrite=FALSE,desiredtz = "Europe/London")

## End(Not run)
```

---

g.readaccfile *Generic functiont to read large blocks of accelerometer data*

Description

The function is used by g.getmeta and g.calibrate to read large blocks of the accelerometer file, which are processed and then deleted. This is needed for memory management.

Usage

```
g.readaccfile(filename,blocksize,blocknumber,
  selectdaysfile=c(),filequality, decn,dayborder,ws,desiredtz=c(),
  PreviousEndPage=1,inspectfileobject=c())
```

Arguments

filename	filename
blocksize	Size of blocks (in file pages) to be read
blocknumber	Block number relative to start of file
selectdaysfile	See documentation g.getmeta
filequality	Single row dataframe with columns: filetooshort, filecorrupt, and filedoesnotholdday. All with the value TRUE or FALSE
decn	Character with a dot or a comma, used for interpretting samplefrequency in the file header. decn is derived with g.dotorcomma
dayborder	See documentation g.part1
ws	Larger windowsize for non-detection, see documentation g.part2
desiredtz	Desired timezone, see documentation g.getmeta
PreviousEndPage	Page number on which previous block ended (automatically assigned within g.getmeta and g.calibrate).
inspectfileobject	Output from the function g.inspectfile.

Value

P=P,filequality=filequality, switchoffLD = switchoffLD

- P Block object extracted from file with format specific to accelerometer brand
- filequality Same as in function arguments
- switchoffLD Boolean to indicate whether it is worth continueing to read the next block of data or not
- endpage Page number on which blocked ends, this will be used as input for argument PreviousEndPage when reading the next block.

Author(s)

Vincent T van Hees <vincentvanhees@gmail.com>

Examples

```
## Not run:
filequality = data.frame(filetooshort=FALSE, filecorrupt=FALSE,
filedoesnotholdday = FALSE)
output = g.readaccfile(filename="C:/myfile.bin",
locksize=20000, blocknumber=1,
selectdaysfile=c(), filequality=filequality,
                        decn=".", dayborder=0,PreviousEndPage=c())

## End(Not run)
```

g.report.part2 *Generate report from milestone data produced by g.part2*

Description

Creates report from milestone data produced by g.part2. Not intended for direct use by package user

Usage

```
g.report.part2(metadatadir=c(), f0=c(), f1=c(), maxdur = 7,
selectdaysfile=c())
```

Arguments

metadatadir	see g.part2
f0	see g.part2
f1	see g.part2
maxdur	see g.part2
selectdaysfile	see g.part2

Value

Function does not produce data, but only writes reports in csv format and visual reports in pdf format

Author(s)

Vincent T van Hees <vincentvanhees@gmail.com>

g.report.part4 *Generate report from milestone data produced by g.part4*

Description

Creates report from milestone data produced by g.part4. Not intended for direct use by package user

Usage

```
g.report.part4(datadir=c(),metadatadir=c(),loglocation = c(),f0=c(),
  f1=c(),storefolderstructure=TRUE)
```

Arguments

datadir	see g.part4
metadatadir	see g.part4
loglocation	see g.part4
f0	see g.part4
f1	see g.part4
storefolderstructure	see g.part4

Value

Function does not produce data, but only writes reports in csv format and a visual report in pdf

Author(s)

Vincent T van Hees <vincentvanhees@gmail.com>

---

g.report.part5 *Generate report from milestone data produced by g.part5*

Description

Creates report from milestone data produced by g.part5. Not intended for direct use by package user

Usage

```
g.report.part5(metadatadir=c(),f0=c(),f1=c(),loglocation=c(),
                           includenightcrit=c(),includedaycrit=c())
```

Arguments

metadatadir	see g.part5
f0	see g.part5
f1	see g.part5
loglocation	see g.part4
includenightcrit	see g.part4
includedaycrit	see g.part2

Value

Function does not produce data, but only writes reports in csv format

Author(s)

Vincent T van Hees <vincentvanhees@gmail.com>

g.shell.GGIR *Shell function for analysing an 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 remember to set argument do.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 = FALSE,
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,
#------------------------------
# Part 5 parameters:
#------------------------------
# Key functions: Merging physical activity with sleep analyses
threshold.lig = c(30,40,50),
threshold.mod = c(100,120),
threshold.vig = c(400,500),
excludefirstlast = FALSE,
```

```
                boutcriter = 0.8,
                boutcriter.in = 0.9,
                boutcriter.lig = 0.8,
                boutcriter.mvpa = 0.8,
                boutdur.in = c(10,20,30),
                boutdur.lig = c(1,5,10),
                boutdur.mvpa = c(1,5,10),
                timewindow = c("WW"),
                #----------------------------------
                # Report generation
                #------------------------------
                do.report=c(2,4,5))

## End(Not run)
```

`g.sib.det` *sustiained inactivty bouts detection*

Description

Detects sustiained inactivty bouts. Function not intended for direct use by package user

Usage

```
g.sib.det(M,IMP,I,twd=c(-12,12),anglethreshold = 5,
                    timethreshold = c(5,10), acc.metric = "ENMO",
                    desiredtz="Europe/London",constrain2range=TRUE)
```

Arguments

`M`	Object produced by g.getmeta
`IMP`	Object produced by g.impute
`I`	Object produced by g.inspectfile
`twd`	Vector of length 2, indicating the time window to consider as hours relative to midnight.
`anglethreshold`	See g.part3
`timethreshold`	See g.part3
`acc.metric`	Which one of the metrics do you want to consider to analyze L5. The metric of interest need to be calculated in M (see g.part1)
`desiredtz`	See g.part3
`constrain2range`	See g.part3

Value

- output = Dataframe for every epoch a classification
- detection.failed = Boolean whether detection failed
- L5list = L5 for every day (defined from noon to noon)

Author(s)

Vincent T van Hees <vincentvanhees@gmail.com>

`g.sib.plot` *Create plot of sustained inactivity bouts*

Description

Function create plot of sustained inactivity bouts for quality check purposes as part of g.part3. Not intended for direct use by package user

Usage

```
g.sib.plot(SLE, M, I, plottitle, nightsperpage=7,
desiredtz="Europe/London")
```

Arguments

`SLE`	Output from g.sib.det
`M`	Output from g.getmeta
`I`	Output from g.inspectfile
`plottitle`	Title to be used in the plot
`nightsperpage`	Number of nights to show per page
`desiredtz`	See g.part3

Value

Function has no output other than the plot

Author(s)

Vincent T van Hees <vincentvanhees@gmail.com>

g.sib.sum *sustiained inactivty bouts detection*

Description

Detects sustiained inactivty bouts. Function not intended for direct use by package user

Usage

```
g.sib.sum(SLE,M,ignorenonwear=FALSE,desiredtz="Europe/London")
```

Arguments

SLE	Output from g.sib.det
M	Object produced by g.getmeta
ignorenonwear	See g.part3
desiredtz	See g.part3

Value

Dataframe with per night and per definition of sustained inactivity bouts the start and end time of each sustained inactivity bout

Author(s)

Vincent T van Hees <vincentvanhees@gmail.com>

g.wavread *function to read .wav files as produced by the accelerometer named 'Axivity'*

Description

For reading the wav accelerometer data as collected with an Axivity accelerometer

Usage

```
g.wavread(wavfile, start = 1, end = 100,units="minutes")
```

Arguments

wavfile	filename (required)
start	start point for reading data, see also units
end	end point for reading data, see also units
units	units used for defining start and end

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
timestamps	local timestamps for rawxyz

Author(s)

Vincent T van Hees <vincentvanhees@gmail.com>

---

g.weardec *Detects whether accelerometer is worn*

Description

Uses the object produced by g.part1 to assess whether the accelerometer was worn

Usage

```
g.weardec(M,wearthreshold,ws2)
```

Arguments

M	Object produced by g.getmeta
wearthreshold	Number of axis that at least need to meet the non-wear criteria
ws2	Large windowsize used in seconds to apply non-wear detection Small window size not needed, because this is inherent to the object M

Value

- r1 Participant id extracted from file
- r2 Night number
- r3 Detected onset of sleep expressed as hours since the previous midnight
- LC fraction of 15 minute windows with more than 5 percent clipping
- LC2 fraction of 15 minute windows with more than 80 percent clipping

Author(s)

Vincent T van Hees <vincentvanhees@gmail.com>

Examples

```
data(data.getmeta)
output = g.weardec(M=data.getmeta,wearthreshold=2,ws2=3600)
```

getFirstTimestamp *Extract first timestamp from GENEActiv file*

Description

Extract first timestamp from GENEActiv file, only used when using the selectdaysfile argument. Function not designed for direct use by package user.

Usage

```
getFirstTimestamp(f, p1)
```

Arguments

f	GENEActiv filename
p1	First value of timestamps object

Value

POSIX object withstarttime

Author(s)

Joe Heywood <j.heywood@ucl.ac.uk>

getfolderstructure *Extracts folderstructure based on data directory.*

Description

Extracts folderstructure based on data directory. This is used when accelerometer files are stored in a hierarchical folder structure and the user likes to have a reference to the exact position in the folder tree, rather than just the filename. Function not intended for direct use by package user.

Usage

```
getfolderstructure(datadir=c(),referencefnames=c())
```

Arguments

datadir	Argument datadir as used in various other functions in GGIR
referencefnames	vector with filename to filter on

Value

List with items: itemfullfilenamesvector with all full paths to the folders including the name of the file itself itemfoldernamevector with only the names of the folder in which each file is stroed (so only the most distal folder in the folder tree)

Examples

```
## Not run:
folderstructure = getfolderstructure(datadir)

## End(Not run)
```

`getStartEnd` *Generate start and end time of a day*

Description

Generate start and end time of a day when working with argument selectdaysfile in g.part1. The user provides a date and a start hour which is used to generate the timestamps of the start hour minutes 5 minutes and the start hour plus 24 hours. Function not designed for direct use by package user.

Usage

```
getStartEnd(d, startHour, outputFormat = "%d/%m/%Y %H:%M:%S",
  tz = "Europe/London")
```

Arguments

`d`	character with date (without time) format
`startHour`	Hour that analysis starts at
`outputFormat`	Characterstring indicating outputFormat
`tz`	Same as desiredtz in g.part1

Value

Data.frame with two columns: a start time five minutes before startHour on day d and an endtime 24 hours after startHour

Author(s)

Joe Heywood <j.heywood@ucl.ac.uk>

Examples

```
startandendtime = getStartEnd(d="20/5/2017", startHour=4)
```

getStartEndNumeric *Generate start and end page of a day*

Description

Generate start and end page of a day when working with argument selectdaysfile in g.part1. The user provides a date and a start hour which is used to generate the pages of the start hour minutes 5 minutes and the start hour plus 24 hours. Function not designed for direct use by package user.

Usage

```
getStartEndNumeric(d, hhr, startHour = 4)
```

Arguments

d	Character with date (without time) format
hhr	GENEActiv::header.info(f) output
startHour	Hour that analysis starts at

Value

Data.frame with two columns: a start page five minutes before startHour on day d and an end page 24 hours after startHour

Author(s)

Joe Heywood <j.heywood@ucl.ac.uk>

Examples

```
## Not run:
hhr = GENEActiv::header.info("C:/myfile.bin")
mystartandendpage = getStartEndNumeric(d="20/5/2017", hhr, startHour = 4)

## End(Not run)
```

identify_levels *Identifies levels of behaviour for g.part5 function.*

Description

Identifies levels of behaviour from acceleratioon and sustained inactivity sibdetection (using angles). Function not intended for direct use by package user.

Usage

```
identify_levels(time,diur,sibdetection,ACC,
                            TRLi,TRMi,TRVi,
                            boutdur.mvpa,boutcriter.mvpa,
                            boutdur.lig,boutcriter.lig,
                            boutdur.in,boutcriter.in,
                            ws3,bout.metric)
```

Arguments

time

diur

sibdetection

ACC

TRLi

TRMi

TRVi

boutdur.mvpa

boutcriter.mvpa

boutdur.lig

boutcriter.lig

boutdur.in

boutcriter.in

ws3,bout.metric

Value

List with items: itemLEVELS itemOLEVELS itemLnames itembc.mvpa itembc.lig itembc.in

Examples

```
## Not run:
levels = identify_levels(time,diur,sibdetection,ACC,
                            TRLi,TRMi,TRVi,
                            boutdur.mvpa,boutcriter.mvpa,
                            boutdur.lig,boutcriter.lig,
                            boutdur.in,boutcriter.in,
                            ws3,bout.metric)

## End(Not run)
```

is.ISO8601 *Check whether character timestamp is in iso8601 format.*

Description

Checks whether timestamp stored in character format is in ISO8601 format or not

Usage

```
is.ISO8601(x)
```

Arguments

x	Timestamps in character format either in ISO8601 or as "yyyy-mm-dd hh:mm:ss".

Examples

```
x ="1980-1-1 18:00:00"
is.ISO8601(x)
```

isfilelist *Checks whether datadir is a directory or a vector with filenames*

Description

Checks whether argument datadir used in various other functions in GGIR is the name of a directory that includes data files or whether it is a vector with the full paths to one or more data files

Usage

```
isfilelist(datadir)
```

Arguments

datadir	Argument datadir as used in various other functions in GGIR

Value

Boolean whether it is a list of files (TRUE) or not (FALSE)

Examples

```
## Not run:
isitafilelist = isfilelist(datadir)

## End(Not run)
```

iso8601chartime2POSIX *Convert iso8601 timestamps to POSIX timestamp*

Description

To avoid ambiguities when sharing and comparing timestamps. All timestamps are expressed in iso8601 format: https://en.wikipedia.org/wiki/ISO_8601 However, to generate plots in R we need to convert them back to POSIX

Usage

```
iso8601chartime2POSIX(x,tz)
```

Arguments

x	Vector of timestamps in iso8601 in character format
tz	Timezone of data collection, e.g. "Europe/London". See List_of_tz_database_time_zones on Wikipedia for full list.

Examples

```
x ="2017-05-07T13:00:00+0200"
tz = "Europe/Amsterdam"
x_converted = iso8601chartime2POSIX(x,tz)
```

is_this_a_dst_night *Check whether the night starting on a calendar date has DST.*

Description

Tests whether the night that follows the input calendar date is a night with day saving time (DST) and on what hour the time moved.

Usage

```
is_this_a_dst_night(calendardate=c(),tz="Europe/London")
```

Arguments

calendardate	Character in the format dd/mm/yyyy
tz	Time zone in "Europe/London" format.

Value

dst_night_or_not
: If value=0 no DST, if value=1 time moved forward, if value=-1 time moved forward

dsthour
: Either the double hour or the hour that was skipped, this differs between countries

Examples

```
test4dst = is_this_a_dst_night("23/03/2014",tz="Europe/London")
```

---

numUnpack *Simple function using Rcpp*

Description

Simple function using Rcpp

Usage

```
numUnpack(pack)
```

Arguments

pack
: vector of integer

Examples

```
## Not run:
numUnpack()

## End(Not run)
```

POSIXtime2iso8601 *Convert POSIX to iso8601 timestamp*

Description

To avoid ambiguities when sharing and comparing timestamps. All timestamps are expressed in iso8601 format: https://en.wikipedia.org/wiki/ISO_8601

Usage

```
POSIXtime2iso8601(x,tz)
```

Arguments

x	Vector of timestamps in POSIX format
tz	Timezone of data collection, e.g. "Europe/London". See https://en.wikipedia.org/wiki/List_of_tz_databas for full list

Author(s)

Vincent T van Hees <vincentvanhees@gmail.com>

Examples

```
## Not run:
x ="2017-05-07 13:15:17 CEST"
tz = "Europe/Amsterdam"
x_converted = POSIXtime2iso8601(x,tz)

## End(Not run)
```

---

resample *Simple function using Rcpp*

Description

Simple function using Rcpp

Usage

```
resample(raw, rawTime, time, stop)
```

Arguments

raw	stop-by-3 matrix with raw values of x, y and z.
rawTime	vector with stop elements of raw time.
time	array with required time points.
stop	number of the last known point in raw and rawTime

Examples

```
## Not run:
resample()

## End(Not run)
```

`updateBlocksize` *Update blocksize of data to be read depending on available memory.*

Description

Function queries available memory to either lower or increase the blocksize used by function g.readaccfile

Usage

```
updateBlocksize(blocksize, bsc_qc)
```

Arguments

`blocksize`	Number of filepages (binary data) or rows (other dataformats).
`bsc_qc`	Data.frame with columns time (timestamp from Sys.time) and size (memory size). This is used for housekeeping in g.calibrate and g.getmeta

Value

List with blocksize and bsc_qc, same format as input, although bsc_qc has one new row.

Index