Package 'GGIR'

October 1, 2022

Type Package

Title Raw Accelerometer Data Analysis

Version 2.8-1

Date 2022-10-01

Maintainer Vincent T van Hees <v.vanhees@accelting.com>

Description A tool to process and analyse data collected with wearable raw acceleration sensors as described in Migueles and colleagues (JMPB 2019), and van Hees and colleagues (JApplPhysiol 2014; PLoSONE 2015). The package has been developed and tested for binary data from 'GENEActiv' <https://activinsights.com/> and GENEA devices (not for sale), .csv-export data from 'Actigraph' <https://actigraphcorp.com> devices, and .cwa and .wav-format data from 'Axivity' <https://axivity.com>. These devices are currently widely used in research on human daily physical activity. Further, the package can handle accelerometer data file from any other sensor brand providing that the data is stored in csv format and has either no header or a two column header. Also the package allows for external function embedding.

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 testthat, covr, knitr, rmarkdown, actilifecounts

Imports data.table, foreach, doParallel, signal, zoo, GENEAread, tuneR, unisensR, ineq, read.gt3x, activityCounts, ActCR, methods, GGIRread

Depends stats, utils, R (>= 3.5.0)

VignetteBuilder knitr

R topics documented:

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

Description

Disclaimer: If you are a new GGIR user then please see package vignette for an introduction to GGIR.

This document is primarily aimed at documenting the functions and their input arguments.

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

You can thank us 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 (e.g. Migueles et al. 2019) in your publications.

Details

Package:	GGIR
Type:	Package
Version:	2.8-1
Date:	2022-10-01
License:	LGPL (>= 2.0, < 3)
Discussion group:	https://groups.google.com/forum/#!forum/rpackageggir

Author(s)

- Vincent T van Hees <v.vanhees@accelting.com> main creator and developer
- Zhou Fang developed calibration algorithm used in function g.calibrate
- Joe Heywood helped develop the functionality to process specific recording days
- Severine Sabia, Mathilde Chen, and Manasa Yerramalla extensively tested and provided feedback on various functions
- Joan Capdevila Pujol helped to improve various functions
- Jairo H Migueles <jairohm@ugr.es> helped to improve various functions
- Matthew R Patterson helped with enhancing the visual report.
- Lena Kushleyeva helped fix bug in sleep detection.
- Taren Sanders helped tidy up the parallel processing functionality

References

- Migueles JH, Rowlands AV, et al. GGIR: A Research Community-Driven Open Source R Package for Generating Physical Activity and Sleep Outcomes From Multi-Day Raw Accelerometer Data. Journal for the Measurement of Physical Behaviour. 2(3) 2019. doi:10.1123/jmpb.2018-0063.
- 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)
```

`applyExtFunction` *Apply external function to acceleration data.*

Description

Applies external function to the raw acceleration data within GGIR. This makes it easier for new algorithms developed to be pilotted on accelerometer data while taking advantage of the existing comprehensive GGIR data management and analysis infrastructure. This function is not for direct interaction by user, please supply object `myfun` to GGIR or g.part1. Object `myfun` is a list as detailed below.

Usage

```
applyExtFunction(data, myfun, sf, ws3, interpolationType=1)
```

Arguments

data	Data data.frame as present internally in g.getmeta. It has at least four columns of which the first is the timestamp followed by the x, y, and z acceleration.
myfun	See details, in short: myfun is a list object that holds the external function to be applied to the data and various parameters to aid in the process.
sf	Sample frequency (Hertz) of the data object
ws3	Short epoch size (first value of windowsizes in g.getmeta).
interpolationType	Integer to indicate type of interpolation to be used when resampling time series (mainly relevant for Axivity sensors), 1=linear, 2=nearest neighbour.

Details

See package vignette for detailed tutorial with examples on how to use the function embedding: https://cran.r-project.org/web/package=GGIR/vignettes/applyExtFunction.pdf Function applyExtFunction is typically not used by the GGIR user directly.

Value

The output of the external algorithm aggregated or repeated to fit the short epoch length of GGIR. Therefore, the short epoch length of GGIR should be a multitude of the resolution of the external function output, or visa versa.

Author(s)

Vincent T van Hees <v.vanhees@accelting.com>

CalcSleepRegularityIndex

Calculates Sleep Regularity Index

Description

Calculates Sleep Regularity Index per day pair proposed by Phillips and colleagues in 2017 expanded with day-pair level estimates.

Usage

```
CalcSleepRegularityIndex(data = c(), epochsize = c(), desiredtz= c())
```

Arguments

data	Data.frame produced by function g.sib.det.
epochsize	Numeric value of epoch size in seconds.
desiredtz	Character with timezone database name, see also g.getmeta

Details

Calculates Sleep Regularity Index per day pair. Absense of missing data is not used as a criteria for calculation. Instead the code asses the fraction of the time for which matching valid data points were found in both days. Later in g.part4 this fraction is used to include or exclude days based on the excludenightcrit criteria it also uses for the other sleep variables. In g.report.part4 these day-level SRI values are stored, but also aggregated across all recording days, all weekend days, and all weekend days, respectively. Therefore, this function is broader in functionality than the algorithm proposed by Phillips and colleagues in 2017.

Value

Data.frame with columns: day (day number); Sleep Regularity Index, which by definition must lie in the range -100 (reversed regularity), to 0 (random pattern), to 100 (perfect regularity); weekday (e.g. Wednesday); frac_valid, number between 0 and 1 indicating the fraction of the 24 hour period for which valid data was available in both the current and the next day, and; date.

Author(s)

Vincent T van Hees <v.vanhees@accelting.com>

References

- Andrew J. K. Phillips, William M. Clerx, et al. Irregular sleep/wake patterns are associated with poorer academic performance and delayed circadian and sleep/wake timing. Scientific Reports. 2017 June 12

---

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)
```

---

check_myfun *Checks myfun object before it is passed to applyExtfunction*

Description

Checks that object myfun is a list and check the elements of the list for: that element names are as expected, that value of each element is of the expected type and length.

Usage

```
check_myfun(myfun, windowsizes)
```

Arguments

myfun	See applyExtFunction
windowsizes	See g.getmeta).

Value

0 if all checkes passed, 1 if one or more checks did not pass. Error message are printed to the console with feedback on which checks did not pass.

Author(s)

Vincent T van Hees <v.vanhees@accelting.com>

check_params *Check default parameters*

Description

Checks parameter objects for class and logical combinations. Called from extract_params. Not intended for direct use by GGIR users.

Usage

```
check_params(params_sleep = c(), params_metrics = c(),
                      params_rawdata = c(), params_247 = c(),
                      params_phyact = c(), params_cleaning = c(),
                       params_output = c(), params_general = c())
```

Arguments

params_sleep	List with sleep parameters
params_metrics	List with parameters related to metrics
params_rawdata	List with parameters related to raw data reading and processing
params_247	List with parameters related to 24/7 behavioural analysis, which includes anything that does not fit with physical activity or sleep research
params_phyact	List with parameters related to physical activity analysis
params_cleaning	List with parameters related to cleaning the time series, including masking and imputation
params_output	List with parameters related to how GGIR stores its output
params_general	List with parameters related to general topics

Value

Lists of updated parameter objects

Author(s)

Vincent T van Hees <v.vanhees@accelting.com>

cosinorAnalyses *Apply cosinor anlaysis and extended cosinor analysis*

Description

Applies cosinor anlaysis from the ActCR package to the time series

Usage

```
cosinorAnalyses(Xi, epochsize = 60, timeOffsetHours = 0)
```

Arguments

Xi	Vector with time series of movement indicators
epochsize	Numeric epochsize in seconds
timeOffsetHours	Numeric time in hours relative to next midnight

Author(s)

Vincent T van Hees <v.vanhees@accelting.com>

createConfigFile *Creates Config File based on variables in function GGIR environment*

Description

Only used inside GGIR. Not intended for direct use by user.

Usage

```
createConfigFile(config.parameters = c())
```

Arguments

config.parameters	List with all arguments used in GGIR.

Author(s)

Vincent T van Hees <v.vanhees@accelting.com>

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(), advanced=FALSE)
```

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
advanced	Boolean to indicate whether to create an advanced sleeplog that also includes logs of nap times and nonwear

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 <v.vanhees@accelting.com>

Examples

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

## End(Not run)
```

extract_params *Extract parameters from input and add them to params*

Description

Extracts parameters separately provided by input and adds them to the params objects. Not intended for direct use by GGIR users.

Usage

```
extract_params(params_sleep = c(), params_metrics = c(),
               params_rawdata = c(), params_247 = c(),
               params_phyact = c(), params_cleaning = c(),
               params_output = c(), params_general = c(), input = c(),
               configfile_csv = c(), params2check = c("sleep", "metrics",
               "rawdata", "247", "phyact",
                                          "cleaning", "output", "general"))
```

Arguments

params_sleep	List with sleep parameters
params_metrics	List with parameters related to metrics
params_rawdata	List with parameters related to raw data reading and processing
params_247	List with parameters related to 24/7 behavioural analysis, which includes anything that does not fit with physical activity or sleep research
params_phyact	List with parameters related to physical activity analysis
params_cleaning	List with parameters related to cleaning the time series, including masking and imputation
params_output	List with parameters related to how GGIR stores its output
params_general	List with parameters related to general topics
input	All objects provided by users
configfile_csv	Csv configuration file
params2check	Character vector to indicate which params objects need to be checked. This allows us to prevent the function from checking params objects that are not used in the context where function extract_params is used.

Value

Lists of updated parameter objects

Author(s)

Vincent T van Hees <v.vanhees@accelting.com>

---

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 <v.vanhees@accelting.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, params_247 = c(), params_phyact = c(),
          quantiletype = 7, includedaycrit = 16,
          idloc = 1, snloc = 1,
          dayborder=0,  desiredtz = "", myfun=c(), acc.metric = 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
params_247	See g.part2
params_phyact	See g.part2
quantiletype	type of quantile function to use (default recommended). For details, see quantile function in STATS package

includedaycrit	See g.part1
idloc	See g.part1
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
dayborder	See g.part1
desiredtz	See g.part1
myfun	External function object to be applied to raw data, see g.getmeta.
acc.metric	Character, see g.part1.
...	Any argument used in the previous version of g.analyse, which will now be used to overrule the arguments specified with the parameter objects.

Value

g.analyse generated two data,franeL

summary	summary for the file that was analysed
daysummary	summary per day for the file that was analysed

These data.frames are used by function g.report.part2 to generate csv reports. An exaplantion of all the columns in the data.frame and subsequent csv reports can be found in the package vignette (Output part 2).

Author(s)

Vincent T van Hees <v.vanhees@accelting.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.analyse.avy` *Function supports g.analyse. Not intended for direct use by user.*

Description

Generatess average day analyses and fills corresponding output matrix, g.analyse.

Usage

```
g.analyse.avday(doquan, averageday, M, IMP, t_TWDI, quantiletype,
                ws3, doiglevels, firstmidnighti, ws2, midnightsi,
                params_247 = c(), qcheck = c(), acc.metric = c(), ...)
```

Arguments

doquan	Boolean whether quantile analysis should be done
averageday	As produced by g.impute
M	As produced by g.getmeta
IMP	As produced by g.impute
t_TWDI	Same as qwindow as described in g.analyse
quantiletype	see g.analyse
ws3	Epoch size in seconds
doiglevels	Boolean to indicate whether iglevels should be calculated
firstmidnighti	see g.detecmidnight
ws2	see g.weardec
midnightsi	see g.detecmidnight
params_247	See g.part2
qcheck	Vector with indicators of when data is valid (value=0) or invalid (value=1).
acc.metric	Character, see g.part1. Here, it is used to decided which acceleration metric to use for IVIS and cosinor analyses.
...	Any argument used in the previous version of g.analyse.avday, which will now be used to overrule the arguments specified with the parameter objects.

Value

InterdailyStability

IntradailyVariability

igfullr_names

igfullr

QUAN

qlevels_names

ML5AD

ML5AD_names

Author(s)

Vincent T van Hees <v.vanhees@accelting.com>

---

g.analyse.perday *Function supports g.analyse. Not intended for direct use by user.*

Description

Generates day specific analyses and fills corresponding output matrix, g.analyse.

Usage

```
g.analyse.perday(ndays, firstmidnighti, time, nfeatures,
                 midnightsi, metashort, averageday,
                 doiglevels, nfulldays,lastmidnight, ws3, ws2, qcheck,
               fname, idloc, sensor.location, wdayname, tooshort, includedaycrit,
                 doquan, quantiletype, doilevels, domvpa,
                 mvpanames, wdaycode, ID,
                 deviceSerialNumber, ExtFunColsi, myfun, desiredtz = "",
                 params_247 = c(), params_phyact = c(),
                 ...)
```

Arguments

ndays	Number of days in file
firstmidnighti	see g.detecmidnight
time	timestamp column from metalong converted to character
nfeatures	estimate of number of variables that need to be stored in the output matrix
midnightsi	see g.detecmidnight
metashort	see g.impute

averageday	As produced by g.impute
doiglevels	Boolean to indicate whether iglevels should be calculated
nfulldays	Number of days between the first and last midnight in the recording
lastmidnight	see g.detecmidnight
ws3	Epoch size in seconds
ws2	see g.weardec
qcheck	vector with zeros and ones for each epoch, respenting the quality check derived with g.impute
fname	RData filename produced by g.part1
idloc	see g.analyse
sensor.location	as produced by g.extractheadervars
wdayname	character with weekdayname
tooshort	0 (file not too short) or 1 (file too short)
includedaycrit	see g.analyse
doquan	Boolean whether quantile analysis should be done
quantiletype	see g.analyse
doilevels	Boolean whether to generate ilevels, see g.analyse
domvpa	Boolean whether to do mvpa analysis
mvpanames	Matrix with 6 columns and 1 row holding the names for the six mvpa variables
wdaycode	Equal to M$wday as produced by g.getmeta
ID	Person Identification number, this can be numeric or character
deviceSerialNumber	As produced by g.extractheadervars
ExtFunColsi	column index of metashort where metric is stored
myfun	External function object to be applied to raw data, see g.getmeta.
desiredtz	see g.part1
params_247	See g.part2
params_phyact	See g.part2
...	Any argument used in the previous version of g.analyse.perday, which will now be used to overrule the arguments specified with the parameter objects.

Value

daysummary	Summary per day for the file that was analysed
ds_names	Variable names in daysummary
windowsummary	Window summary, only used when selectdayfile is specified
ws_names	Variable names in windowsummary

Author(s)

Vincent T van Hees <v.vanhees@accelting.com>

g.analyse.perfile *Function supports g.analyse. Not intended for direct use by user.*

Description

Generates recording specific analyses and fills corresponding output matrix, g.analyse.

Usage

```
g.analyse.perfile(ID, fname, deviceSerialNumber,
  sensor.location, startt, I, LC2, LD, dcomplscore,
  LMp, LWp, C, lookat, AveAccAve24hr,
  colnames_to_lookat, QUAN,  ML5AD,
  ML5AD_names, igfullr, igfullr_names,
  daysummary, ds_names, includedaycrit, strategy, hrs.del.start,
  hrs.del.end, maxdur, windowsizes, idloc, snloc, wdayname, doquan,
  qlevels_names, doiglevels, tooshort, InterdailyStability,
  IntradailyVariability,
  IVIS_windowsize_minutes, qwindow, longitudinal_axis_id, cosinor_coef)
```

Arguments

ID	Person Identification number, this can be numeric or character
fname	see g.analyse.perday
deviceSerialNumber	As produced by g.extractheadervars
sensor.location	as produced by g.extractheadervars
startt	First timestamp in metalong
I	output g.inspectfile
LC2	see g.impute
LD	length data in minutes
dcomplscore	see g.impute
LMp	length measurement based on study protocol (minutes)
LWp	length of sensor worn based on study protocol (minutes)
C	output g.calibrate
lookat	indices of metashort column to analyse
AveAccAve24hr	Average acceleration in an average 24 hour cycle
colnames_to_lookat	Names of columns to look at, corresponding to argurment lookat
QUAN	Results quantile analysis on the average day produced by g.analyse.avday
ML5AD	Results ML5 analyses on the average day produced by g.analyse.avday

ML5AD_names	Columns names corresponding to ML5AD
igfullr	Results intensity gradient (ig) analysis on the average day produced by g.analyse.avday
igfullr_names	Columns names corresponding to igfullr
daysummary	object produced by g.analyse.perday
ds_names	column names corresponding to daysummary
includedaycrit	see g.analyse
strategy	see g.analyse
hrs.del.start	see g.analyse
hrs.del.end	see g.analyse
maxdur	see g.analyse
windowsizes	see g.getmeta
idloc	see g.analyse
snloc	see g.analyse
wdayname	character with weekdayname
doquan	Boolean whether quantile analysis should be done
qlevels_names	object produced by g.analyse.avday
doiglevels	Boolean to indicate whether iglevels should be calculated
tooshort	0 (file not too short) or 1 (file too short)
InterdailyStability	see g.IVIS
IntradailyVariability	see g.IVIS
IVIS_windowsize_minutes	see g.IVIS
qwindow	see g.analyse
longitudinal_axis_id	Index of axis for which the angle correlates most strongly across 24 hoursas calculated inside g.analyse. For hip worn accelerometer this helps to check which axis was the veritcal axis. The estimate may not be informative for other attachment locations.
cosinor_coef	output from cosinorAnlayses passed on to be included in file summary

Value

filesummary	summary for the file that was analysed
daysummary	Summary per day for the file that was analysed

Author(s)

Vincent T van Hees <v.vanhees@accelting.com>

`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(data, sf, ws3, metrics2do,
                        n = 4, lb = 0.2, hb = 15,
                        zc.lb = 0.25, zc.hb = 3,
                        zc.sb = 0.01, zc.order = 2,
                        actilife_LFE = FALSE)
```

Arguments

`data`	Three column matrix with x, y, and z acceleration data
`n`	filter order, see GGIR for details
`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
`lb`	Lower boundery of cut-off frequencies, see GGIR.
`hb`	Higher boundery of cut-off frequencies, see GGIR.
`zc.lb`	See GGIR
`zc.hb`	See GGIR
`zc.sb`	See GGIR
`zc.order`	See GGIR
`actilife_LFE`	See GGIR

Value

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

Author(s)

Vincent T van Hees <v.vanhees@accelting.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)
data = cbind(Gx, Gy, Gz)
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,
do.lfx=FALSE, do.lfy=FALSE, do.lfz=FALSE,
do.hfx=FALSE, do.hfy=FALSE, do.hfz=FALSE,
do.bfx=FALSE, do.bfy=FALSE, do.bfz=FALSE,
do.zcx=FALSE, do.zcy=FALSE, do.zcz=FALSE,
do.brondcounts=FALSE, do.neishabouricounts=FALSE)

extractedmetrics = g.applymetrics(data,n=4,sf=40,ws3=5,metrics2do)
```

---

g.binread

Deprecated, internally replaced by function readGenea from the GGIRread package

---

Description

Function g.binread has been deprecated and has been moved to R package GGIRread and renamed readGenea

Usage

```
g.binread(...)
```

Arguments

...	Arguments ignored

---

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, params_rawdata = c(), params_general = c(),
            params_cleaning = c(), ...)
```

Arguments

datafile	Name of accelerometer file
params_rawdata	See g.part1
params_general	See g.part1
params_cleaning	See g.part1
...	Any argument used in the previous version of g.calibrate, which will now be used to overrule the arguments specified with the parameter objects.

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 where temperate data is collected and available to GGIR, such as GENEActiv, Axivity, and in some instances ad-hoc .csv data.

Author(s)

Vincent T van Hees <v.vanhees@accelting.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.conv.actlog` *Function to read activity log and make it useful for the rest of GGIR.*

Description

Function to read activity log and convert it into data.frame that has for each ID and date a different qwindow vector

Usage

```
g.conv.actlog(qwindow, qwindow_dateformat="%d-%m-%Y")
```

Arguments

`qwindow`	Path to csv file with activity log. Expected format of the activity diary is: First column headers followed by one row per recording, first column is recording ID, which needs to match with the ID GGIR extracts from the accelerometer file. Followed by date column in format "23-04-2017", where date format is specified by argument qwindow_dateformat (below). Use the character combination date, Date or DATE in the column name. This is followed by one or multiple columns with start times for the activity types in that day format in hours:minutes:seconds. The header of the column will be used as label for each activity type. Insert a new date column before continuing with activity types for next day. Leave missing values empty. If an activitylog is used then individuals who do not appear in the activitylog will still be processed with value c(0,24). Dates with no activiy log data can be skipped, no need to have a column with the date followed by a column with the next date.
`qwindow_dateformat`	Character specifying the date format used in the activity log.

Value

Data.frame with column ID, date and qwindow, where each qwindow value is a qwindow vector

Author(s)

Vincent T van Hees <v.vanhees@accelting.com>

g.convert.part2.long *Convert part 2 report to long format*

Description

Not for direct access by used. This function is used inside g.report.part2 and convert2 part 2 report to long ormat if there are multiple segments per day

Usage

```
g.convert.part2.long(daySUMMARY)
```

Arguments

daySUMMARY	Object available inside g.report.part2

Value

Data.frame with long format version of daySUMMARY

Author(s)

Vincent T van Hees <v.vanhees@accelting.com>

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 <v.vanhees@accelting.com>

---

g.cwaread — *Deprecated, internally replaced by function readAxivity from the GGIRread package*

Description

Function g.cwaread has been deprecated and has been moved to R package GGIRread and renamed readAxivity

Usage

```
g.cwaread(...)
```

Arguments

...	Arguments ignored

---

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

Description

Detect all midnights in a time series

Usage

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

Arguments

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

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 <v.vanhees@accelting.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 = "", loadGENEActiv = "GGIRread", ...)
```

Arguments

`inputfile`	full path to inputfile
`dformat`	Data format code: 1=.bin, 2=.csv, 3=.wav, 4=.cwa, 5=.csv for ad-hoc monitor brand
`mon`	Monitor code (accelorometer brand): 0=undefined, 1=GENEA, 2=GENEActiv, 3=Actigraph, 4=Axivity, 5=Movisense, 6=Verisense
`desiredtz`	Desired timezone, see documentation g.getmeta
`loadGENEActiv`	See GGIR
`...`	Any input arguments needed for function read.myacc.csv if you are working with a non-standard csv formatted files.

Value

Character object showing how decimals are separated

Author(s)

Vincent T van Hees <v.vanhees@accelting.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 <v.vanhees@accelting.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
- BodyLocation = Attachement location of the sensor
- SX = sex
- deviceSerialNumber = serial number

Author(s)

Vincent T van Hees <v.vanhees@accelting.com>

Examples

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

g.fragmentation *Fragmentation metrics from time series.*

Description

The function is used by g.part5 to derive time series fragmentation metrics. The function assumes that NA values and nonwear time is accounted for before the data enters the function.

Usage

```
g.fragmentation(frag.metrics = c("mean", "TP", "Gini", "power",
      "CoV", "NFragPM", "all"), LEVELS = c(), Lnames=c(), xmin=1)
```

Arguments

frag.metrics	Character with fragmentation metric to exract. Can be "mean", "TP", "Gini", "power", or "CoV", "NFragPM", or all the above metrics with "all". See details.
LEVELS	Numeric vector of behavioural level classes derived with identify_levels
Lnames	Character vector with names of classes used in LEVELS, see details.
xmin	Numeric scalar to indicate the minimum recordable fragment length. In g.part5 this is derived from the epoch length.

Details

See package vignette for description of fragmentation metrics. In short, abbreviation "TP" refers to transition probality metrics, abbreviation "CoV" refers to Coefficient of Variance, and metric "NFragPM" refers to the Number of fragments per minute.

Regarding the Lnames argument. The class names included in this are categorised as follows:

- Inactive - if name includes the character strings "day_IN_unbt" or "day_IN_bts".
- LIPA - If name includes the character strings "day_LIG_unbt" or "day_LIG_bts".
- MVPA - If name includes the character strings "day_MOD_unbt", "day_VIG_unbt", or "day_MVPA_bts"

Value

List with Character object showing how decimals are separated

TP_PA2IN	Transition probability physical activity to inactivity
TP_IN2PA	Transition probability physical inactivity to activity
Nfrag_IN2LIPA	Number of inacitivty fragments succeeded by LIPA (light physical activity)
TP_IN2LIPA	Transition probability physical inactivity to LIPA
Nfrag_IN2MVPA	Number of inacitivty fragments succeeded by MVPA (moderate or vigorous physical activity)
TP_IN2MVPA	Transition probability physical inactivity to MVPA
Nfrag_MVPA	Number of MVPA fragments
Nfrag_LIPA	Number of LIPA fragments
mean_dur_MVPA	mean MVPA fragment duration
mean_dur_LIPA	mean LIPA fragment duration
Nfrag_IN	Number of inactivity fragments
Nfrag_PA	Number of activity fragments
mean_dur_IN	mean duration inactivity fragments
mean_dur_PA	mean duration activity fragments
Gini_dur_IN	Gini index corresponding to inactivity fragment durations
Gini_dur_PA	Gini index corresponding to activity fragment durations
CoV_dur_IN	Coefficient of Variance corresponding to inactivity fragment durations
CoV_dur_PA	Coefficient of Variance corresponding to activity fragment durations

alpha_dur_IN	Alpha of the fitted power distribution through inactivity fragment durations
alpha_dur_PA	Alpha of the fitted power distribution through activity fragment durations
x0.5_dur_IN	x0.5 corresponding to alpha_dur_IN
x0.5_dur_PA	x0.5 corresponding to alpha_dur_PA
W0.5_dur_IN	W0.5 corresponding to alpha_dur_IN
W0.5_dur_PA	W0.5 corresponding to alpha_dur_PA
NFragPM_IN	Number of IN fragments per minutes in IN
NFragPM_PA	Number of PA fragments per minutes in PA
SD_dur_IN	Standard deviation in the duration of inactivity fragments
SD_dur_PA	Standard deviation in the duration of physical activity fragments

Author(s)

Vincent T van Hees <v.vanhees@accelting.com>

Examples

```
## Not run:
    x = c(6, 5, 6, 7, 6, 6, 7, 6, 6, 5, 6, 6, 6, 5, 7, 6, 6, 5, 5, 5, 6, 7, 6,
        6, 6, 6, 7, 6, 5, 5, 5, 5, 5, 6, 6, 6, 5, 6, 6, 6, 6, 6, 6, 6, 6, 6,
        7, 7, 6, 5, 6, 5, 6, 5, rep(12, 11), 5, 6, 6, 6, 5, 6, rep(9, 14), 6,
        5, 7, 7, 6, 7, 7, 7, 6, 6, 6, 5, 6, 5, 5, 5, 6, 5, 5, 5, 5, 5, 5, 5)
  Lnames = c("spt_sleep", "spt_wake_IN", "spt_wake_LIG", "spt_wake_MOD",
             "spt_wake_VIG", "day_IN_unbt", "day_LIG_unbt", "day_MOD_unbt",
             "day_VIG_unbt", "day_MVPA_bts_10", "day_IN_bts_30",
              "day_IN_bts_10_30", "day_LIG_bts_10")
  out = g.fragmentation(frag.metrics = "all",
                        LEVELS = x,
                        Lnames=Lnames)
## End(Not run)
```

`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=6, 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 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 may simplify 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 larger than 1 minute and with fraction of time larger than the boutcriter threshold. If value=4, same as 3 but also requires the first and last epoch to meet the threshold criteria. If value=5, same as 4, but now looks for breaks larger than a minute such that 1 minute breaks are allowe, and the fraction of time that meets the threshold should be equal than or greater than the bout.criter threshold. If value=6, algorithm improved (2021) to check for first and last epoch.
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 <v.vanhees@accelting.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.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,qM5L5=c(),
iglevels=c(), MX.ig.min.dur=10)
```

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
qM5L5	Percentiles (quantiles) to be calculated over L5 and M5 window.
iglevels	See g.analyse. If provided then the intensity gradient will be calculated for all MX windows larger or equal than argument MX.ig.min.dur
MX.ig.min.dur	Minimum MX duration needed in order for intensity gradient to be calculated

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 <v.vanhees@accelting.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 bin files. Further, the function should be able to cope with ActiGraph gt3x and csv files, Axivity cwa and csv files, Movisens bin files, and ad-hoc csv files read through the read.myacc.csv function.

Usage

```
g.getmeta(datafile, params_metrics = c(), params_rawdata = c(),
                    params_general = c(), daylimit = FALSE,
                offset = c(0, 0, 0), scale = c(1, 1, 1), tempoffset = c(0, 0, 0),
                    meantempcal = c(), myfun = c(), ...)
```

Arguments

`datafile`	name of accelerometer file
`params_metrics`	See g.part1
`params_rawdata`	See g.part1
`params_general`	See g.part1
`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)
`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)
`myfun`	External function object to be applied to raw data. See details applyExtFunction.
`...`	Any argument used in the previous version of g.getmeta, which will now be used to overrule the arguments specified with the parameter objects.

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 <v.vanhees@accelting.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,
configtz = NULL)
```

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
configtz	Same as in g.dotorcomma

Value

The starttime

Author(s)

Vincent T van Hees <v.vanhees@accelting.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, params_cleaning = c(),
desiredtz="", dayborder= 0, TimeSegments2Zero =c(), ...)
```

Arguments

M	output from g.getmeta
I	output from g.inspectfile
params_cleaning	See g.part1
desiredtz	See g.part1

dayborder See g.part1

TimeSegments2Zero
Optional data.frame to specify which time segments need to be ignored for the imputation, and acceleration metrics to be imputed by zeros. The data.frame is expected to contain two columns named windowstart and windowend, with the start- and end time of the time segment in POSIXlt class.

... Any argument used in the previous version of g.impute, which will now be used to overrule the arguments specified with the parameter objects.

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 <v.vanhees@accelting.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.imputeTimegaps *Impute gaps in three axis raw accelerometer data*

Description

Removes all sample with a zero in each of the three axes, and then (as default) imputes time gaps by the last recorded value per axis normalised to 1 _g_

Usage

```
g.imputeTimegaps(x, xyzCol, timeCol = c(), sf, k = 0.25, impute = TRUE,
                 PreviousLastValue = c(0,0,1),
                 PreviousLastTime = NULL, epochsize = NULL)
```

Arguments

x	Data.frame with raw accelerometer data, and a timestamp column with millisecond resolution.
xyzCol	Columnnames or numbers for the x, y and z column
timeCol	Column name or number for the timestamp column
sf	Sample frequency in Hertz
k	Minimum time gap length to be imputed
impute	Boolean to indicate whether the time gaps identified should be imputed
PreviousLastValue	Automatically identified last value in previous chunk of data read.
PreviousLastTime	Automatically identified last timestamp in previous chunk of data read.
epochsize	Numeric vector of length two, with short and long epoch sizes.

Value

Data.frame based on input x with timegaps imputed (as default) or with recordings with 0 values in the three axes removed (if impute = FALSE)

Author(s)

Vincent T van Hees <v.vanhees@accelting.com>

---

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 = "", params_rawdata = c(),
                        configtz = c(), ...)
```

Arguments

datafile	name of data file
desiredtz	Desired timezone, see documentation g.getmeta
params_rawdata	See g.part1
configtz	...
...	Any argument used in the previous version of g.getmeta, which will now be used to overrule the arguments specified with the parameter objects.

Value

header	fileheader
monn	monitor name (genea, geneactive)
monc	monitor brand code (0 - ad-hoc file format, 1 = genea (non-commercial), 2 = GENEActive, 3 = actigraph, 4 = Axivity (AX3, AX6), 5 = Movisense, 6 = Verisense)
dformn	data format name, e.g bin, csv, cwa, gt3x
dformc	data format code (1 = .bin, 2 = .csv, 3 = .wav, 4 = .cwa, 5 = ad-hoc .csv, 6 = .gt3x)
sf	samplefrequency in Hertz
filename	filename

Author(s)

Vincent T van Hees <v.vanhees@accelting.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 <v.vanhees@accelting.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.IVIS *Calculates IV and IS*

Description

To extract interdaily stability and interdaily variability as originally proposed by van Someren.

Usage

```
g.IVIS(Xi, epochsizesecondsXi = 5, IVIS_epochsize_seconds = c(),
  IVIS_windowsize_minutes = 60, IVIS.activity.metric = 1,
  IVIS_acc_threshold = 20, IVIS_per_daypair = FALSE)
```

Arguments

Xi	Vector with acceleration values, e.g. ENMO metric.
epochsizesecondsXi	Epoch size of the values in Xi expressed in seconds.
IVIS_epochsize_seconds	This argument has been depricated.
IVIS_windowsize_minutes	Window size of the Intradaily Variability (IV) and Interdaily Stability (IS) metrics in minutes, needs to be able to add up to 24 hours.
IVIS.activity.metric	Metric used for activity calculation. Value = 1, uses continuous scaled acceleration. Value = 2, tries to collapse acceleration into a binary score of rest versus active to try to simulate the original approach.
IVIS_acc_threshold	Acceleration threshold to distinguish inactive from active
IVIS_per_daypair	Boolean to indicate whether IVIS should be calculated per day pair and then aggregated across day pairs weighted by day completeness (default FALSE).

Value

`InterdailyStability`

`IntradailyVariability`

Author(s)

Vincent T van Hees <v.vanhees@accelting.com>

References

- Eus J. W. Van Someren, Dick F. Swaab, Christopher C. Colenda, Wayne Cohen, W. Vaughn McCall & Peter B. Rosenquist. Bright Light Therapy: Improved Sensitivity to Its Effects on Rest-Activity Rhythms in Alzheimer Patients by Application of Nonparametric Methods Chronobiology International. 1999. Volume 16, issue 4.

Examples

```
Xi = abs(rnorm(n = 10000,mean = 0.2))
IVISvariables = g.IVIS(Xi=Xi)
```

---

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, sleeplogsep=",", meta.sleep.folder = c(),
desiredtz="")
```

Arguments

`loglocation`	Location of the spreadsheet (csv) with sleep log information. See package vignette for explanation on expected format
`coln1`	Column number in the sleep log spreadsheet where the onset of the first night starts
`colid`	Column number in the sleep log spreadsheet in which the participant ID code is stored (default = 1)
`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)?

`sleeplogsep` Value used as sep argument for reading sleeplog csv file, usually "," or ";".

`meta.sleep.folder` Path to part3 milestone data, only specify if sleeplog is in advanced format.

`desiredtz` See g.part4

Value

Data frame with sleeplog, which can be either in basic format or in advanced format. See GGIR package vignette for discussion of these two formats.

Author(s)

Vincent T van Hees <v.vanhees@accelting.com>

Examples

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

## End(Not run)
```

---

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 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(),
      studyname = c(), myfun = c(), params_metrics = c(), params_rawdata = c(),
        params_cleaning = c(), params_general = c(), ...)
```

Arguments

datadir	Directory where the accelerometer files are stored, e.g. "C:/mydata", or list 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.
f0	File index to start with (default = 1). Index refers to the filenames sorted in alphabetical order
f1	File index to finish with (defaults to number of files available, i.e., f1 = 0)
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
myfun	External function object to be applied to raw data. See details applyExtFunction.
params_metrics	See details in GGIR.
params_rawdata	See details in GGIR.
params_cleaning	See details in GGIR.
params_general	See details in GGIR.
...	If you are working with a non-standard csv formatted files, g.part1 also takes any input arguments needed for function read.myacc.csv and argument rmc.noise from get_nw_clip_block_params. First test these argument with function read.myacc.csv directly. To ensure compatibility with R scripts written for older GGIR versions, the user can also provide parameters listed in the params_ objects as direct argument.

Details

GGIR comes with many processing parameters, which have been thematically grouped in parameter objects (R list). By running print(load_params()) you can see the default values of all the parameter objects. When g.part 1 is used via GGIR you have the option to specifiy a configuration file, which will overrule the default parameter values. Further, as user you can set parameter values as input argument to both g.part1 and GGIR. Directly specified argument overrule the configuration file and default values.

See the GGIR package vignette or the details section in GGIR for a more elaborate overview of parameter objects and their usage across GGIR.

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 <v.vanhees@accelting.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 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(),
      myfun = c(), params_cleaning = c(), params_247 = c(),
      params_phyact = c(), params_output = c(), params_general = c(), ...)
```

Arguments

`datadir`	Directory where the accelerometer files are stored, e.g. "C:/mydata", or list of accelerometer filenames and directories, e.g. c("C:/mydata/myfile1.bin", "C:/mydata/myfile2.bin").
`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 GGIR will recognise what the value of metadatadir is.
`f0`	File index to start with (default = 1). Index refers to the filenames sorted in alphabetical order

f1	File index to finish with (defaults to number of files available, i.e., f1 = 0)
myfun	External function object to be applied to raw data. See details applyExtFunction.
params_cleaning	See details in GGIR.
params_247	See details in GGIR.
params_phyact	See details in GGIR.
params_output	See details in GGIR.
params_general	See details in GGIR.
...	To ensure compatibility with R scripts written for older GGIR versions, the user can also provide parameters listed in the params_ objects as direct argument.

Details

GGIR comes with many processing parameters, which have been thematically grouped in parameter objects (R list). By running print(load_params()) you can see the default values of all the parameter objects. When g.part 2 is used via GGIR you have the option to specifiy a configuration file, which will overrule the default parameter values. Further, as user you can set parameter values as input argument to both g.part2 and GGIR. Directly specified argument overrule the configuration file and default values.

See the GGIR package vignette or the details section in GGIR for a more elaborate overview of parameter objects and their usage across GGIR.

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 <v.vanhees@accelting.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 function 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 GGIR only.

Usage

```
g.part3(metadatadir=c(), f0, f1, myfun=c(),
  params_sleep = c(), params_metrics = c(), params_output = c(), params_general = c(),
  ...)
```

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 GGIR will recognise what the value of metadatadir is.
f0	File index to start with (default = 1). Index refers to the filenames sorted in alphabetical order
f1	File index to finish with (defaults to number of files available, i.e., f1 = 0)
myfun	External function object to be applied to raw data. See details applyExtFunction.
params_sleep	See details in GGIR.
params_metrics	See details in GGIR.
params_output	See details in GGIR.
params_general	See details in GGIR.
...	To ensure compatibility with R scripts written for older GGIR versions, the user can also provide parameters listed in the params_ objects as direct argument.

Details

GGIR comes with many processing parameters, which have been thematically grouped in parameter objects (R list). By running print(load_params()) you can see the default values of all the parameter objects. When g.part 3 is used via GGIR you have the option to specifiy a configuration file, which will overrule the default parameter values. Further, as user you can set parameter values as input argument to both g.part3 and GGIR. Directly specified argument overrule the configuration file and default values.

See the GGIR package vignette or the details section in GGIR for a more elaborate overview of parameter objects and their usage across GGIR.

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 <v.vanhees@accelting.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 part of the Sleep Period Time window or not*

Description

Combines output from g.part3 and guider information to estimate sleep variables. See vignette paragraph "Sleep and full day time-use analysis in GGIR" for an elaborate descript of the sleep detection.

Usage

```
g.part4(datadir = c(), metadatadir = c(), f0 = f0, f1 = f1, params_sleep = c(),
   params_metrics = c(),  params_cleaning = c(), params_output = c(),
                  params_general = c(), ...)
```

Arguments

datadir	Directory where the accelerometer files are stored, e.g. "C:/mydata", or list of accelerometer filenames and directories, e.g. c("C:/mydata/myfile1.bin", "C:/mydata/myfile2.bin").
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 GGIR will recognise what the value of metadatadir is.
f0	File index to start with (default = 1). Index refers to the filenames sorted in alphabetical order
f1	File index to finish with (defaults to number of files available, i.e., f1 = 0)
params_sleep	List of parameters used for sleep analysis (GGIR part 3, 4, and 5): see documentation g.part3.
params_metrics	List of parameters used for metrics extraction (GGIR part 1): see documentation g.part1.
params_cleaning	See details in GGIR.
params_output	See details in GGIR.
params_general	See details in GGIR.
...	To ensure compatibility with R scripts written for older GGIR versions, the user can also provide parameters listed in the params_ objects as direct argument.

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. See package vignette paragraph "Output part 4" for description of all the variables.

Author(s)

Vincent T van Hees <v.vanhees@accelting.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.part4_extractid *Extracts ID from filename and finds matching rows in sleeplog*

Description

Extracts ID from filename and finds matching rows in sleeplog. Function not designed for direct use by GGIR users.

Usage

```
g.part4_extractid(idloc, fname, dolog, sleeplogidnum, sleeplog, accid = c())
```

Arguments

idloc	See g.part4
fname	Full patth to filename
dolog	Boolean to indicate whether to rely on a sleeplog
sleeplogidnum	Should the participant identifier as stored in the sleeplog be interpretted as a number (TRUE=default) or a character (FALSE)?
sleeplog	Sleeplog data.frame passed on from g.part4
accid	ID extracted from the acceleration file in GGIR part3. If not available leave blank.

Value

List with accid the ID and matching_indices_sleeplog a vector with matching row indices in the sleeplog

Author(s)

Vincent T van Hees <v.vanhees@accelting.com>

`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(),
                    params_sleep = c(), params_metrics = c(),
                    params_247 = c(), params_phyact = c(),
                    params_cleaning = c(), params_output = c(),
                    params_general = c(), ...)
```

Arguments

`datadir`	Directory where the accelerometer files are stored, e.g. "C:/mydata", or list of accelerometer filenames and directories, e.g. c("C:/mydata/myfile1.bin", "C:/mydata/myfile2.bin").
`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 GGIR will recognise what the value of metadatadir is.
`f0`	File index to start with (default = 1). Index refers to the filenames sorted in alphabetical order
`f1`	File index to finish with (defaults to number of files available, i.e., f1 = 0)
`params_sleep`	See details in GGIR.
`params_metrics`	See details in GGIR.
`params_247`	See details in GGIR.
`params_phyact`	See details in GGIR.
`params_cleaning`	See details in GGIR.
`params_output`	See details in GGIR.
`params_general`	See details in GGIR.
`...`	To ensure compatibility with R scripts written for older GGIR versions, the user can also provide parameters listed in the params_ objects as direct argument.

Value

The function does not produce values but generates an RData file in the milestone subfolder ms5.out which incudes a dataframe named `output`. This dataframe is used in g.report.part5 to create two reports one per day and one per person. See package vignette paragraph "Output part 5" for description of all the variables.

Author(s)

Vincent T van Hees <v.vanhees@accelting.com>

Examples

```
## Not run:
  metadatadir = "C:/myfolder/meta"
  g.part5(metadatadir=metadatadir)

## End(Not run)
```

`g.part5.addfirstwake` *Adds first wake if it is missing in part 4 output.*

Description

Not intended for direct use by GGIR users. Adds first wake if it is missing in part 4 output as part of g.part5.

Usage

```
g.part5.addfirstwake(ts, summarysleep_tmp2, nightsi, sleeplog,
ID, Nepochsinhour, Nts, SPTE_end, ws3new)
```

Arguments

`ts`
`summarysleep_tmp2`

`nightsi`

`sleeplog`

`ID`

`Nepochsinhour`

`Nts`

`SPTE_end`

`ws3new`

Value

Data.frame ts

Author(s)

Vincent T van Hees <v.vanhees@accelting.com>

`g.part5.addsib` *Adds the sustained inactivity bout to the ts series.*

Description

Not intended for direct use by GGIR users. Adds the sustained inactivity bout to the ts series as part of g.part5.

Usage

```
g.part5.addsib(ts,ws3, Nts, S2, desiredtz, j, nightsi)
```

Arguments

`ts`

`ws3`

`Nts`

`S2`

`desiredtz`

`j`

`nightsi`

Value

Data.frame ts

Author(s)

Vincent T van Hees <v.vanhees@accelting.com>

`g.part5.classifyNaps` *Classify Naps from identified sustained inactivty bouts*

Description

Classify Naps from identified sustained inactivty bouts, based on model that was originally trained with hip-worn accelerometer data in 3-3.5 year olds. Assume that metric ENMO is used and HASIB.algo is set to vanHees2015.

Usage

```
g.part5.classifyNaps(sibreport = c(), desiredtz = "",
      possible_nap_window = c(9, 18),
      possible_nap_dur = c(15, 240),
      nap_model = "hip3yr", HASIB.algo = "vanHees2015")
```

Arguments

`sibreport`	Object generated by g.sibreport
`desiredtz`	See g.getmeta.
`possible_nap_window`	Numeric vector of length two with range in clock hours during which naps are assumed to take place.
`possible_nap_dur`	Numeric vector of length two with range in duration (minutes) of a nap.
`nap_model`	Character to specify classification model. Currently the only option is "hip3yr", which corresponds to a model trained with hip data in 3-3.5 olds trained with parent diary data.
`HASIB.algo`	See g.part3.

Value

Data.frame with classified naps and newly detected non-wear periods.

Author(s)

Vincent T van Hees <v.vanhees@accelting.com>

`g.part5.definedays` *Fix missing night in part 4 output*

Description

Not intended for direct use by GGIR users. Defines when day windows start and end as part of g.part5.

Usage

```
g.part5.definedays(nightsi, wi, indjump, nightsi_bu,
                       ws3new, qqq_backup=c(), ts, Nts, timewindowi, Nwindows)
```

Arguments

`nightsi`

`wi`

`indjump`

`nightsi_bu`

`ws3new`

`qqq_backup`

`ts`

`Nts`

`timewindowi`

`Nwindows`

Value

List of qqq and qqq_backup

Author(s)

Vincent T van Hees <v.vanhees@accelting.com>

g.part5.fixmissingnight

Fix missing night in part 4 output

Description

Not intended for direct use by GGIR users. If a night is missing in the part4 output then this function tries to fix as part of g.part5.

Usage

```
g.part5.fixmissingnight(summarysleep_tmp2, sleeplog=c(), ID)
```

Arguments

summarysleep_tmp2	Object produced by g.part4
sleeplog	
ID	

Value

Corrected summarysleep_tmp2 object.

Author(s)

Vincent T van Hees <v.vanhees@accelting.com>

g.part5.handle_lux_extremes

Check lux values for extremes and imputes or removes them

Description

Extreme values are imputed by mean of neightbours if they occur isolated or in a sequence of two, and removed if they occure in a sequence of 3 or longer.

Usage

```
g.part5.handle_lux_extremes(lux)
```

Arguments

lux	Vector with lux values

Value

List of imputed lux values and a vector with matching length named correction_log indicating which timestamps where imputed (value=1), replaced by NA (value=2) or untouched (value=0).

Author(s)

Vincent T van Hees <v.vanhees@accelting.com>

`g.part5.lux_persegment` *Extract key lux variables per segment of the data.*

Description

Extracts per segment of the day: mean lux, time above 1000 lux, time awake, and time LUX imputed. Function not intended for direct use by package user.

Usage

```
g.part5.lux_persegment(ts, sse, LUX_day_segments, ws3new)
```

Arguments

`ts`

`sse`

`LUX_day_segments`

`ws3new`

Value

List with values (vector) of the derived variables and corresponding names (vector).

Author(s)

Vincent T van Hees <v.vanhees@accelting.com>

g.part5.onsetwaketiming

Identify wake and sleepperiod window timing

Description

Not intended for direct use by GGIR users. Labels timing of wakeing up and sleep onset as part of g.part5.

Usage

```
g.part5.onsetwaketiming(qqq, ts, min, sec, hour, timewindowi, skiponset, skipwake)
```

Arguments

qqq

ts

min

sec

hour

timewindowi

skiponset

skipwake

Value

A list with objects: wake, onset, wakei, onseti, skiponset, and skipwake.

Author(s)

Vincent T van Hees <v.vanhees@accelting.com>

g.part5.savetimeseries

Saves oart 5 time series to csv files

Description

Not intended for direct use by GGIR users. Saves oart 5 time series to csv files as part of g.part5.

Usage

```
g.part5.savetimeseries(ts, LEVELS, desiredtz, rawlevels_fname,
save_ms5raw_format="csv",
save_ms5raw_without_invalid=TRUE,
DaCleanFile=c(), includedaycrit.part5=2/3, ID=c())
```

Arguments

ts	
LEVELS	
desiredtz	See g.getmeta.
rawlevels_fname	
save_ms5raw_format	See g.part5
save_ms5raw_without_invalid	See g.part5
DaCleanFile	Content of data_cleaning_file as documented in g.report.part5. Only used in this function if save_ms5rawlevels is TRUE, and it only affects the time series files stored.
includedaycrit.part5	See g.report.part5. Only used in this function if save_ms5rawlevels is TRUE, and it only affects the time series files stored.
ID	If data_cleaning_file is used then this argument specifies which participant ID the data correspond with.

Value

Function does not provide output, it only prepare data for saving and saves it to a file.

Author(s)

Vincent T van Hees <v.vanhees@accelting.com>

---

g.part5.wakesleepwindows

Label wake and sleepperiod window

Description

Not intended for direct use by GGIR users. Label wake and sleepperiod window as part of g.part5.

Usage

```
g.part5.wakesleepwindows(ts, summarysleep_tmp2, desiredtz,
nightsi, sleeplog, ws3new, Nts, ID, Nepochsinhour)
```

Arguments

ts	data.frame with time series
summarysleep_tmp2	cleaned output from part 4
desiredtz	
nightsi	
sleeplog	
ws3new	
Nts	
ID	
Nepochsinhour	

Value

Object ts

Author(s)

Vincent T van Hees <v.vanhees@accelting.com>

---

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 <v.vanhees@accelting.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. The first part of the report summarizes important daily metrics in bar plot format. The second part of the report shows the raw data and annotations in 24-hr periods. Angle-z is shown with sleep annotations during the SPT (sleep period time) window. ENMO is shown with daytime inactivity and PA (physical activity) annotations in the lower section of each 24-hr plot. The PA annotations are based on a 10 minute bout metric and 80 of a 10 minute bout of MVPA. Vigorous PA is a short window of time above threshold.vig that is part of a bout of MVPA. Light PA is a short window of time above threshold.lig that is part of a bout of light PA.*

Description

Function called by 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",
 threshold.lig = 30, threshold.mod = 100, threshold.vig = 400)
```

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 GGIR will recognise what the value of metadatadir is.
dofirstpage	Boolean to indicate whether a first page with historgrams summarizing the whole measurement should be added
viewingwindow	See GGIR
f0	File index to start with (default = 1). Index refers to the filenames sorted in alphabetical order
f1	File index to finish with (defaults to number of files available, i.e., f1 = 0)
overwrite	See 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
threshold.lig	See g.part5
threshold.mod	See g.part5
threshold.vig	See g.part5

Value

No values, this function only generates a plot

Author(s)

Vincent T van Hees <v.vanhees@accelting.com> Matthew R Patterson <mpatterson@shimmersensing.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",
threshold.lig,threshold.mod,threshold.vig)

## 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 from memory. This is needed for memory management.

Usage

```
g.readaccfile(filename, blocksize, blocknumber, filequality,
                        decn, ws, PreviousEndPage = 1, inspectfileobject = c(),
                         PreviousLastValue = c(0,0,1), PreviousLastTime = NULL,
                           params_rawdata = c(), params_general = c(), ...)
```

Arguments

filename	filename
blocksize	Size of blocks (in file pages) to be read
blocknumber	Block number relative to start of file
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
ws	Larger windowsize for non-detection, see documentation g.part2
PreviousEndPage	Page number on which previous block ended (automatically assigned within g.getmeta and g.calibrate).
inspectfileobject	Output from the function g.inspectfile.
PreviousLastValue	Automatically identified last value in previous chunk of data read.
PreviousLastTime	Automatically identified last timestamp in previous chunk of data read.
params_rawdata	See g.part1
params_general	See g.part1
...	Any input arguments needed for function read.myacc.csv if you are working with a non-standard csv formatted files. Furter, any argument used in the previous version of g.readaccfile, which will now be used to overrule the arguments specified with the parameter objects.

Value

- 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 <v.vanhees@accelting.com>

Examples

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

## End(Not run)
```

---

g.readtemp_movisens *Reads the temperature from movisens files.*

Description

Reads the temperature from movisens files, resamples it and adds it to the matrix where accelerations are stored

Usage

```
g.readtemp_movisens(datafile, desiredtz = "", from = c(), to = c(),
  interpolationType=1)
```

Arguments

datafile	Full path to the folder where the movisens bin files are stored. Note that movisens store a set of bin file in one folder per recording. GGIR will read the pertinent bin file to access to the temperature data.
desiredtz	See g.getmeta
from	Origin point to derive the temperature from movisens files (automatically calculated by GGIR)
to	End point to derive the temperature from movisens files (automatically calculated by GGIR)
interpolationType	Integer to indicate type of interpolation to be used when resampling time series (mainly relevant for Axivity sensors), 1=linear, 2=nearest neighbour.

Value

Data matrix with the temperature values resampled at 64 Hz.

Examples

```
## Not run:
  P = g.readtemp_movisens(datafile, desiredtz = "", from = c(), to = 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 = 0,
selectdaysfile=c(), store.long=FALSE, do.part2.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 GGIR will recognise what the value of metadatadir is.
f0	File index to start with (default = 1). Index refers to the filenames sorted in alphabetical order
f1	File index to finish with (defaults to number of files available, i.e., f1 = 0)
maxdur	see g.part2
selectdaysfile	see g.part2
store.long	Booelean to indicate whether output should stored in long format in addition to default wide format. Automatically turned to TRUE if using day segmentation with qwindow.
do.part2.pdf	Boolean, 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 <v.vanhees@accelting.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, data_cleaning_file=c(), sleepwindowType = "SPT")
```

Arguments

datadir	Directory where the accelerometer files are stored, e.g. "C:/mydata", or list of accelerometer filenames and directories, e.g. c("C:/mydata/myfile1.bin", "C:/mydata/myfile2.bin").
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 GGIR will recognise what the value of metadatadir is.
loglocation	see g.part4
f0	File index to start with (default = 1). Index refers to the filenames sorted in alphabetical order
f1	File index to finish with (defaults to number of files available, i.e., f1 = 0)
storefolderstructure	see g.part4
data_cleaning_file	see g.part4
sleepwindowType	see g.part4

Value

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

The following files are stored in the root of the results folder: part4_nightsummary_sleep_cleaned.csv part4_summary_sleep_cleaned.csv

The following files are stored in the folder results/QC: part4_nightsummary_sleep_full.csv part4_summary_sleep_full.csv

If a sleeplog is used *_full.csv as stored in the QC folder includes estimates for all nights in the data, and *_cleaned.csv in the results folder includes estimates for all nights in the data excluding the nights that did not had a sleeplog entry or had no valid accelerometer data.

If a sleep log is not used then * _cleaned.csv includes the nights that are in *_full.csv excluding the nights with insufficient data.

If you have a study where the sleeplog was available for a subset of the participants, but you want to include all individuals in your analysis, then use the *_full.csv output and clean the night level

data yourself by excluding rows with cleaningcode > 1 which are the cases where no or invalid accelerometer data was present.

The above means that for studies with missing sleeplog entries for some individuals and some nights using the *_full.csv output and excluding rows (nights) with cleaningcode > 1 will lead to the same as * _cleaned.csv plus sleep estimates for the nights with missing sleeplog, providing that there was enough accelerometer data for those nights.

In other words, *_cleaned.csv is perfect if you only want to rely on nights with a sleeplog or if you do not use a sleeplog at all. For all other scenarios We advise using the *_full.csv report and to clean it yourself.

See package vignette sections "Sleep analysis" and "Output part 4" for a more elaborative description of the sleep analysis and reporting.

Author(s)

Vincent T van Hees <v.vanhees@accelting.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(),
                           data_cleaning_file=c(),
                           includedaycrit.part5=2/3,
                           minimum_MM_length.part5=23,
                           week_weekend_aggregate.part5=FALSE,
                           LUX_day_segments=c())
```

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 GGIR will recognise what the value of metadatadir is.
`f0`	File index to start with (default = 1). Index refers to the filenames sorted in alphabetical order
`f1`	File index to finish with (defaults to number of files available, i.e., f1 = 0)
`loglocation`	see g.part4
`includenightcrit`	Depricated as of version 2.0, not used anymore in part 5 report

`includedaycrit` Depricated as of version 2.0, not used anymore in part 5 report

`data_cleaning_file`
see g.part4

`includedaycrit.part5`
Inclusion criteria for number of valid hours, either as expressed as a ratio of 1 or as the number of hours in a 24 hour day.

`minimum_MM_length.part5`
Minimum length in hours of a MM day to be included in the cleaned part 5 results.

`week_weekend_aggregate.part5`
Boolean to indicate whether week and weekend-days aggregates should be stored. This is turned off by default as it generates a large number of extra columns in the output report.

`LUX_day_segments`
see g.part5

Value

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

The following files are stored in the root of the results folder: part5_daysummary_* part5_personsummary_*

The following files are stored in the folder results/QC: part5_daysummary_full_*

See package vignette paragraph "Waking-waking or 24 hour time-use analysis" and "Output part 5" for a more elaborative description of the full day time-use and analysis and reporting.

Author(s)

Vincent T van Hees <v.vanhees@accelting.com>

---

g.shell.GGIR *Wrapper function around function GGIR*

Description

This function used to be the central function in the package, but has been renamed GGIR. You can still use function call g.shell.GGIR but all arguments will be passed on to function GGIR. We have done this to preserve consistency with older use cases of the GGIR package. All documentation can now be found in GGIR.

Usage

```
g.shell.GGIR(...)
```

Arguments

`...` Any of the parameters used by GGIR.

Value

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

Author(s)

Vincent T van Hees <v.vanhees@accelting.com>

`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),
          acc.metric = "ENMO", desiredtz = "",
      myfun=c(), sensor.location = "wrist", params_sleep = c(), zc.scale = 1, ...)
```

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.
`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
`myfun`	External function object to be applied to raw data. See details applyExtFunction.
`sensor.location`	Character to indicate sensor location, default is wrist. If it is hip HDCZA algorithm also requires longitudinal axis of sensor to be between -45 and +45 degrees.
`params_sleep`	See g.part3
`zc.scale`	Used for zero-crossing counts only. Scaling factor to be applied after counts are calculated (GGIR part 3). See GGIR.
`...`	Any argument used in the previous version of g.sib.det, which will now be used to overrule the arguments specified with the parameter objects.

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 <v.vanhees@accelting.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="")
```

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 <v.vanhees@accelting.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=TRUE,desiredtz="")
```

Arguments

SLE	Output from g.sib.det
M	Object produced by g.getmeta
ignorenonwear	If TRUE then ignore detected monitor non-wear periods to avoid confusion between monitor non-wear time and sustained inactivity (default = TRUE)
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 <v.vanhees@accelting.com>

g.sibreport *Generate sustiained inactivty bouts report*

Description

Generate sustained inactivity bout report. Function not intended for direct use by package user

Usage

```
g.sibreport(ts, ID, epochlength, logs_diaries=c(), desiredtz="")
```

Arguments

ts	Data frame with time series as created inside function g.part5
ID	Recording identifier (character or numeric)
epochlength	Numeric to indicate epoch length in seconds in the ts object
logs_diaries	Object produced by g.loadlog function
desiredtz	See g.getmeta

Value

Dataframe with one row per sustained inactivity bout and corresponding properties stored in the data.frame columns.

Author(s)

Vincent T van Hees <v.vanhees@accelting.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 <v.vanhees@accelting.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 <v.vanhees@accelting.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)
```

`get_nw_clip_block_params`

Set monitor brand specific parameters

Description

Set monitor brand specific thresholds for non-wear detection, clipping etection, and blocksizes to be loaded. Not designed for direct use by user.

Usage

```
get_nw_clip_block_params(chunksize, dynrange, monc, rmc.noise=c(),
sf, dformat, rmc.dynamic_range)
```

Arguments

`chunksize`	See g.calibrate
`dynrange`	See g.getmeta
`monc`	See g.inspectfile
`rmc.noise`	Noise level of acceleration signal in _g_-units, used when working ad-hoc .csv data formats using read.myacc.csv. The read.myacc.csv does not take rmc.noise as argument, but when interacting with GGIR or g.part1 rmc.noise is used. There, rmc.noise is taken from the params_rawdata object if not explicitly specified by user.
`sf`	Numeric, sample frequency in Hertz
`dformat`	See g.dotorcomma
`rmc.dynamic_range`	Optional, please see read.myacc.csv

Author(s)

Vincent T van Hees <v.vanhees@accelting.com>

get_starttime_weekday_meantemp_truncdata
Get starttime (adjusted), weekday, mean temp, and adjust data accordingly.

Description

Function not intended for direct use by user. Used inside g.getmeta as an intermediate step between loading the raw data and calibrating it. This step includes extracting the starttime and adjusting it to nearest integer number of long epoch window lengths in an hour, truncating the data accordingly, extracting the corresponding weekday and mean temperature (if temperature is available).

Usage

```
get_starttime_weekday_meantemp_truncdata(temp.available, monc,
dformat, data, P, header, desiredtz, sf, i,
datafile, ws2, starttime, wday, weekdays, wdayname, configtz = NULL)
```

Arguments

temp.available	Boolean whether temperate is available.
monc	See g.inspectfile
dformat	See g.dotorcomma
data	Data part of g.readaccfile output
P	data loaded from accelerometer file with g.readaccfile
header	Header part of g.readaccfile output
desiredtz	See g.getmeta
sf	Numeric, sample frequency in Hertz
i	Integer index of passed on from g.getmeta to indicate what data block is being read.
datafile	See g.getmeta
ws2	Long epoch length
starttime	Once calculate it is remembered and fed into this function again, such that it does not have to be recalulated.
wday	Once calculate it is remembered and fed into this function again, such that it does not have to be recalulated.
weekdays	Once calculate it is remembered and fed into this function again, such that it does not have to be recalulated.
wdayname	Once calculate it is remembered and fed into this function again, such that it does not have to be recalulated.
configtz	See g.getmeta

Author(s)

Vincent T van Hees <v.vanhees@accelting.com>

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, and produces user-friendly reports. The function acts as a shell with calls to g.part1, g.part2, g.part3, g.part4 and g.part5.

Usage

```
GGIR(mode = 1:5,
     datadir = c(),
     outputdir = c(),
     studyname = c(),
     f0 = 1, f1 = 0,
     do.report = c(2, 4, 5),
     configfile = c(),
     myfun = c(), ...)
```

Arguments

mode	Numeric (default = 1:5). Specify which of the five parts need to be run, e.g., mode = 1 makes that g.part1 is run; or mode = 1:5 makes that the whole GGIR pipeline is run, from g.part1 to g.part5.
datadir	Character (default = c()). Directory where the accelerometer files are stored, e.g., "C:/mydata", or list of accelerometer filenames and directories, e.g. c("C:/mydata/myfile1.bin", "C:/mydata/myfile2.bin").
outputdir	Character (default = c()). 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	Character (default = c()). 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	Numeric (default = 1). File index to start with (default = 1). Index refers to the filenames sorted in alphabetical order.
f1	Numeric (default = 0). File index to finish with (defaults to number of files available).
do.report	Numeric (default = c(2, 4, 5)). For which parts to generate a summary spreadsheet: 2, 4, and/or 5. Default is c(2, 4, 5). 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.

`configfile` Character (default = c()). Configuration file previously generated by function GGIR. See details.

`myfun` List (default = c()). External function object to be applied to raw data. See package vignette for detailed tutorial with examples on how to use the function embedding: https://cran.r-project.org/package=GGIR/vignettes/ExternalFunction.pdf

`...` Any of the parameters used GGIR. Given the large number of parameters used in GGIR we have grouped them in objects that start with "params_". These are documented in the details section. You cannot provide these objects as argument to function GGIR, but you can provide the parameters inside them as input to function GGIR.

Details

Once you have used function GGIR and the output directory (outputdir) will be filled with milestone data and results. Function GGIR stores all the explicitely entered argument values and default values for the argument that are not explicitely provided in a csv-file named config.csv stored in the root of the output folder. The config.csv file is accepted as input to GGIR with argument `configfile` to replace the specification of all the arguments, except `datadir` and `outputdir`.

The practical value of this is that it eases the replication of analysis, because instead of having to share you R script, sharing your config.csv file will be sufficient. Further, the config.csv file contribute to the reproducibility of your data analysis.

Note: When combining a configuration file with explicitely provided argument values, the explicitely provided argument values will overrule the argument values in the configuration file. If a parameter is neither provided via the configuration file nor as input then GGIR uses its default paramter values which can be inspected with command `print(load_params())`, and if you are specifically interested in a certain subgroup of parameters, e.g., physical activity, then you can do `print(load_params()$params_phyact)`. These defaults are part of the GGIR code and cannot be changed by the user.

The parameters that can be used in GGIR are:

params_general: A list of parameters used across all GGIR parts that do not fall in any of the other categories.

overwrite Boolean (default = FALSE). 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.

selectdaysfile Character (default =). Do not use, this is legacy code for one specific data study. 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 Numeric (default = 0). Hour at which days start and end (dayborder = 4 would mean 4 am).

do.parallel Boolean (default = TRUE). Whether to use multi-core processing (only works if at least 4 CPU cores are available).

maxNcores Numeric (default maxNcores=). Maximum number of cores to use when argument do.parallel is set to true. GGIR by default uses either the maximum number of available cores or the number of files to process (whichever is lower), but this argument allows you to set a lower maximum.

acc.metric Boolean (default = ENMO). 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).

part5_agg2_60seconds Boolean (default = FALSE). Wether to use aggregate epochs to 60 seconds as part of the GGIR g.part5 analysis.

print.filename Boolean (default = FALSE). Whether to print the filename before before analysing it (in case do.parallel = FALSE). Printing the filename can be useful to investigate problems (e.g., to verify that which file is being read).

desiredtz Character (default = "", i.e., system timezone). Timezone in which device was configured and experiments took place. If experiments took place in a different timezone, then use this argument for the timezone in which the experiments took place and argument `configtz` to specify where the device was configured. See also https://en.wikipedia.org/wiki/Zone.tab

configtz Character (default = "", i.e., system timezone). Only functional for AX3 cwa and ActiGraph .gt3x data at the moment. Timezone in which the accelerometer was configured. Only use this argument if the timezone of configuration and timezone in which recording took place are different. See also https://en.wikipedia.org/wiki/Zone.tab

sensor.location Character (default = "wrist"). To indicate sensor location, default is wrist. If it is hip, the HDCZA algorithm for sleep detection also requires longitudinal axis of sensor to be between -45 and +45 degrees.

windowsizes Numeric vector, three values (default = c(5, 900, 3600)). To indicate the lengths of the windows as in c(window1, window2, window3): window1 is the short epoch length in seconds, by default 5, and 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 (expected to be a multitude of 60 seconds); 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.

idloc Numeric (default = 1). If idloc = 1 the code assumes that ID number is stored in the obvious header field. Note that for ActiGraph data the ID is never stored in the file header. For value set to 2, 5, 6, and 7, GGIR looks at the filename and extracts the character string preceding the first occurance of a "_" (idloc = 2), " " (space, idloc = 5), "." (dot, idloc = 6), and "-" (idloc = 7), respectively. You may have noticed that idloc 3 and 4 are skipped, they were used for one study in 2012, and not actively maintained anymore, but because it is legacy code not omitted.

expand_tail_max_hours Numeric (default = 0). Number hours to expand the g.part1 output with synthetic data to trigger sleep detection for last night. The synthetic data for metashort entails: timestamps continuing regularly, zeros for acceleration metrics other than EN, one for EN. Angle columns are created in a way that it triggers the sleep detection using the equation: `round(sin((1:length_expansion) / (900/epochsize))) * 15`. To keep track of the tail expansion g.part1 stores the lenght of the expansion in the RData files, which is then passed via g.part2, g.part3, and g.part4 to g.part5. In g.part5 it is then included as an additional variable in the csv-reports. In the g.part4 report the last nigth is omitted, because we know that sleep estimates from the last night will not be trustworthy. In the g.part5 output most columns related to the sleep assessment will be omitted for the last window to avoid

biassing the averages. Using argument `expand_tail_max_hours` implies the assumption that the participant fell asleep at or before the end of the recording if the recording ended less than `expand_tail_max_hours` before midnight. This assumption may not always hold true and should be used with caution.

params_rawdata: A list of parameters used to related to reading and pre-processing raw data, excluding parameters related to metrics as those are in the params_metrics object.

backup.cal.coef Character (default = "retrieve"). Option to use backed-up calibration coefficient instead of deriving the calibration coefficients when analysing the same file twice. Argument backup.cal.coef has two usecase. Use case 1: 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. This can be useful for analysing short lasting laboratory experiments with insufficient sphere data to perform the auto-calibration, but for which calibration coefficients can be derived in an alternative way. It is the users responsibility to compile the csv-spreadsheet. Instead of building this file the user can also Use case 2: The user wants to avoid performing the auto-calibration repeatedly on the same file. If backup.cal.coef value is set to "retrieve" (default) then GGIR will look out for the "data_quality_report.csv" file in the outputfolder QC, which holds the previously generated calibration coefficients. If you do not want this happen, then deleted the data_quality_report.csv from the QC folder or set it to value "redo".

minimumFileSizeMB Numeric (default = 2). Minimum File size in MB required to enter processing. This argument can help to avoid having short uninformative files to enter the analyses. Given that a typical accelerometer collects several MBs per hour, the default setting should only skip the very tiny files.

do.cal Boolean (default = TRUE). Whether to apply auto-calibration or not by g.calibrate. Recommended setting is TRUE.

imputeTimegaps Boolean (default = TRUE). To indicate whether timegaps larger than 1 sample should be imputed. Currently only used for .gt3x data and ActiGraph .csv format, where timegaps can be expected as a result of Actigraph's idle sleep.mode configuration.

spherecrit Numeric (default = 0.3). 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 Numeric (default = 72). 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 Boolean (default = FALSE). If TRUE will print a summary of the calibration procedure in the console when done.

chunksize Numeric (default = 1). Value 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, 1 equals 12 hour chunks. For machines with less than 4Gb of RAM memory a value below 1 is recommended.

dynrange Numeric (default = NULL). Provide dynamic range for accelerometer data to overwrite hardcoded 6 g for GENEA and 8 g for other brands.

interpolationType Integer (default = 1). To indicate type of interpolation to be used when re-sampling time series (mainly relevant for Axivity sensors), 1=linear, 2=nearest neighbour.

rmc.file Character (default = NULL). Filename of file to be read if it is in the working directory, or full path to the file otherwise.

rmc.nrow Numeric (default = NULL). Number of rows to read, same as nrow argument in read.csv and nrows in fread. The whole file is read by default (i.e., rmc.nrow = Inf).

rmc.skip Numeric (default = NULL). Number of rows to skip, same as skip argument in read.csv and in fread.

rmc.dec Character (default = "."). Decimal used for numbers, same as dec argument in read.csv and in fread.

rmc.firstrow.acc Numeric (default = NULL). First row (number) of the acceleration data.

rmc.firstrow.header Numeric (default = NULL). First row (number) of the header. Leave blank if the file does not have a header.

rmc.header.length Numeric (default = NULL). If file has header, specify header length (number of rows).

rmc.col.acc Numeric, three values (default = c(1, 2, 3)). Vector with three column (numbers) in which the acceleration signals are stored.

rmc.col.temp Numeric (default = NULL). Scalar with column (number) in which the temperature is stored. Leave in default setting if no temperature is avaible. The temperature will be used by g.calibrate.

rmc.col.time Numeric (default = NULL). Scalar with column (number) in which the timestamps are stored. Leave in default setting if timestamps are not stored.

rmc.unit.acc Character (default = "g"). Character with unit of acceleration values: "g", "mg", or "bit".

rmc.unit.temp Character (default = "C"). Character with unit of temperature values: (K)elvin, (C)elsius, or (F)ahrenheit.

rmc.unit.time Character (default = "POSIX"). Character with unit of timestamps: "POSIX", "UNIXsec" (seconds since origin, see argument `rmc.origin`), "character", or "ActivPAL" (exotic timestamp format only used in the ActivPAL activity monitor).

rmc.format.time Character (default = "%Y-%m-%d %H:%M:%OS"). Character giving a date-time format as used by strptime. Only used for rmc.unit.time: character and POSIX.

rmc.bitrate Numeric (default = NULL). If unit of acceleration is a bit then provide bit rate, e.g., 12 bit.

rmc.dynamic_range Numeric or character (default = NULL). If unit of acceleration is a bit then provide dynamic range deviation in g from zero, e.g., +/-6g would mean this argument needs to be 6. If you give this argument a character value the code will search the file header for elements with a name equal to the character value and use the corresponding numeric value next to it as dynamic range.

rmc.unsignedbit Boolean (default = TRUE). If unsignedbit = TRUE means that bits are only positive numbers. if unsignedbit = FALSE then bits are both positive and negative.

rmc.origin Character (default = "1970-01-01"). Origin of time when unit of time is UNIXsec, e.g., 1970-1-1.

rmc.desiredtz Character (default = "", i.e., system timezone). Timezone in which device was configured and experiments took place. If experiments took place in a different timezone, then use this argument for the timezone in which the experiments took place and argument configtz to specify where the device was configured (not implemented yet). See also https://en.wikipedia.org/wiki/Zone.tab

rmc.sf Numeric (default = NULL). Sample rate in Hertz, if this is stored in the file header then that will be used instead (see argument `rmc.headername.sf`).

rmc.headername.sf Character (default = NULL). If file has a header: Row name under which the sample frequency can be found.

rmc.headername.sn Character (default = NULL). If file has a header: Row name under which the serial number can be found.

rmc.headername.recordingid Character (default = NULL). If file has a header: Row name under which the recording ID can be found.

rmc.header.structure Character (default = NULL). Used to split the header name from the header value, e.g., ":" or " ".

rmc.check4timegaps Boolean (default = FALSE). To indicate whether gaps in time should be imputed with zeros. Some sensing equipment provides accelerometer with gaps in time. The rest of GGIR is not designed for this, by setting this argument to TRUE the gaps in time will be filled with zeros.

rmc.col.wear Numeric (default = NULL). If external wear detection outcome is stored as part of the data then this can be used by GGIR. This argument specifies the column in which the wear detection (Boolean) is stored.

rmc.doresample Boolean (default = FALSE). To indicate whether to resample the data based on the available timestamps and extracted sample rate from the file header.

rmc.noise Numeric (default = 13). Noise level of acceleration signal in m*g*-units, used when working ad-hoc .csv data formats using read.myacc.csv. The read.myacc.csv does not take rmc.noise as argument, but when interacting with GGIR or g.part1 rmc.noise is used.

loadGENEActiv Character (default = GGIRread). To indicate which package should be used to read GENEActiv .bin files: either GENEAread or GGIRread

params_metrics: A list of parameters used to specify the signal metrics that need to be extract in GGIR g.part1.

do.anglex Boolean (default = FALSE). If TRUE, calculates the angle of the X axis relative to the horizontal:

$$angleX = (\tan^{-1} \frac{acc_{rollmedian(x)}}{acc_{rollmedian(y)} + acc_{rollmedian(z)}}) * 180/\pi$$

do.angley Boolean (default = FALSE). If TRUE, calculates the angle of the Y axis relative to the horizontal:

$$angleY = (\tan^{-1} \frac{acc_{rollmedian(y)}}{acc_{rollmedian(x)} + acc_{rollmedian(z)}}) * 180/\pi$$

do.anglez Boolean (default = TRUE). If TRUE, calculates the angle of the Z axis relative to the horizontal:

$$angleZ = (\tan^{-1} \frac{acc_{rollmedian(z)}}{acc_{rollmedian(x)} + acc_{rollmedian(y)}}) * 180/\pi$$

do.zcx Boolean (default = FALSE). If TRUE, calculates metric zero-crossing count for x-axis. For computation specifics see source code of function g.applymetrics

do.zcy Boolean (default = FALSE). If TRUE, calculates metric zero-crossing count for y-axis. For computation specifics see source code of function g.applymetrics

do.zcz Boolean (default = FALSE). If TRUE, calculates metric zero-crossing count for z-axis. For computation specifics see source code of function g.applymetrics

do.enmo Boolean (default = TRUE). If TRUE, calculates the metric:

$$ENMO = \sqrt{acc_x^2 + acc_y^2 + acc_z^2} - 1$$

(if ENMO < 0, then ENMO = 0).

do.lfenmo Boolean (default = FALSE). If TRUE, calculates the metric ENMO over the low-pass filtered accelerations (for computation specifics see source code of function g.applymetrics). The filter bound is defined by the parameter hb.

do.en Boolean (default = FALSE). If TRUE, calculates the Euclidean Norm of the raw accelerations:

$$EN = \sqrt{acc_x^2 + acc_y^2 + acc_z^2}$$

do.mad Boolean (default = FALSE). If TRUE, calculates the Mean Amplitude Deviation:

$$MAD = \frac{1}{n}\Sigma|r_i - \overline{r}|$$

do.enmoa Boolean (default = FALSE). If TRUE, calculates the metric:

$$ENMOa = \sqrt{acc_x^2 + acc_y^2 + acc_z^2} - 1$$

(if ENMOa < 0, then ENMOa = |ENMOa|).

do.roll_med_acc_x Boolean (default = FALSE). If TRUE, calculates the metric. For computation specifics see source code of function g.applymetrics.

do.roll_med_acc_y Boolean (default = FALSE). If TRUE, calculates the metric. For computation specifics see source code of function g.applymetrics.

do.roll_med_acc_z Boolean (default = FALSE). If TRUE, calculates the metric. For computation specifics see source code of function g.applymetrics.

do.dev_roll_med_acc_x Boolean (default = FALSE). If TRUE, calculates the metric. For computation specifics see source code of function g.applymetrics.

do.dev_roll_med_acc_y Boolean (default = FALSE). If TRUE, calculates the metric. For computation specifics see source code of function g.applymetrics.

do.dev_roll_med_acc_z Boolean (default = FALSE). If TRUE, calculates the metric. For computation specifics see source code of function g.applymetrics.

do.bfen Boolean (default = FALSE). If TRUE, calculates the metric. For computation specifics see source code of function g.applymetrics.

do.hfen Boolean (default = FALSE). If TRUE, calculates the metric. For computation specifics see source code of function g.applymetrics.

do.hfenplus Boolean (default = FALSE). If TRUE, calculates the metric. For computation specifics see source code of function g.applymetrics.

do.lfen Boolean (default = FALSE). If TRUE, calculates the metric. For computation specifics see source code of function g.applymetrics.

do.lfx Boolean (default = FALSE). If TRUE, calculates the metric. For computation specifics see source code of function g.applymetrics.

do.lfy Boolean (default = FALSE). If TRUE, calculates the metric. For computation specifics see source code of function g.applymetrics.

do.lfz Boolean (default = FALSE). If TRUE, calculates the metric. For computation specifics see source code of function g.applymetrics.

do.hfx Boolean (default = FALSE). If TRUE, calculates the metric. For computation specifics see source code of function g.applymetrics.

do.hfy Boolean (default = FALSE). If TRUE, calculates the metric. For computation specifics see source code of function g.applymetrics.

do.hfz Boolean (default = FALSE). If TRUE, calculates the metric. For computation specifics see source code of function g.applymetrics.

do.bfx Boolean (default = FALSE). If TRUE, calculates the metric. For computation specifics see source code of function g.applymetrics.

do.bfy Boolean (default = FALSE). If TRUE, calculates the metric. For computation specifics see source code of function g.applymetrics.

do.bfz Boolean (default = FALSE). If TRUE, calculates the metric. For computation specifics see source code of function g.applymetrics.

do.brondcounts Boolean (default = FALSE). If TRUE, calculates the metric via R package activityCounts. We call them BrondCounts because there are large number of acitivty counts in the physical activity and sleep research field. By calling them _brondcounts_ we clarify that these are the counts proposed by Jan Brønd and implemented in R by Ruben Brondeel. The _brondcounts_ are intended to be an imitation of the counts produced by one of the closed source ActiLife software by ActiGraph.

do.neishabouricounts Boolean (default = FALSE). If TRUE, calculates the metric via R package actilifecounts, which is an implementation of the algorithm used in the closed-source software ActiLife by ActiGraph (methods published in doi: 10.1038/s41598-022-16003-x). We use the name of the first author (instead of ActiLifeCounts) of the paper and call them NeishabouriCount under the uncertainty that ActiLife will implement this same algorithm over time.

lb Numeric (default = 0.2). Lower boundary of the frequency filter (in Hertz) as used in the filter-based metrics.

hb Numeric (default = 15). Higher boundary of the frequency filter (in Hertz) as used in the filter-based metrics.

n Numeric (default = 4). Order of the frequency filter as used in the filter-based metrics.

zc.lb default = 0.25) Used for zero-crossing counts only. Lower boundery of cut-off frequency filter.

zc.hb default = 3) Used for zero-crossing counts only. Higher boundery of cut-off frequencies in filter.

zc.sb default = 0.01) Stop band used for calculation of zero crossing counts. Value is the acceleration threshold in g units below which acceleration will be rounded to zero.

zc.order default = 2) Used for zero-crossing counts only. Order of frequency filter.

zc.scale default = 1) Used for zero-crossing counts only. Scaling factor to be applied after counts are calculated (GGIR part 3).

actilife_LFE Boolean (default = FALSE). If TRUE, calculates the NeishabouriCount metric with the low-frequency extension filter as proposed in the closed source ActiLife software by ActiGraph. Only applicable to the metric NeishabouriCount.

params_cleaning: A list of parameters used across all GGIR parts releated to masking or imputing data, abbreviated as "cleaning".

do.imp Boolean (default = TRUE). Whether to impute missing values (e.g., suspected of monitor non-wear or clippling) or not by g.impute in GGIR g.part2. Recommended setting is TRUE.

TimeSegments2ZeroFile Data frame (default = NULL). Optional data.frame to specify which time segments need to be ignored for the imputation, and acceleration metrics to be imputed by zeros. The data.frame is expected to contain two columns named windowstart and windowend, with the start- and end time of the time segment in POSIXlt class.

data_cleaning_file Character (default = NULL). Optional path to a csv file you create that holds four columns: ID, day_part5, relyonguider_part4, and night_part4. ID should hold the participant ID. Columns day_part5 and night_part4 allow you to specify which day(s) and night(s) need to be excluded from g.part5 and g.part4, respectively. So, this will be done regardless of whether the rest of GGIR thinks those day(s)/night(s) are valid. Column relyonguider_part4 allows you to specify for which nights g.part4 should fully rely on the guider. See also package vignette.

excludefirstlast.part5 Boolean (default = FALSE). If TRUE then the first and last window (waking-waking or midnight-midnight) are ignored in g.part5.

excludefirstlast Boolean (default = FALSE). If TRUE then the first and last night of the measurement are ignored for the sleep assessment in g.part4.

excludefirst.part4 Boolean (default = FALSE). If TRUE then the first night of the measurement are ignored for the sleep assessment in g.part4.

excludelast.part4 Boolean (default = FALSE). If TRUE then the last night of the measurement are ignored for the sleep assessment in g.part4.

includenightcrit Numeric (default = 16). Minimum number of valid hours per night (24 hour window between noon and noon), used for sleep assessment in g.part4.

minimum_MM_length.part5 Numeric (default = 23). Minimum length in hours of a MM day to be included in the cleaned g.part4 results.

selectdaysfile Numeric (default = NULL). Functionality designed for the London Centre of Longidutinal studies. Csv file holding the relation between device serial numbers and measurement days of interest.

strategy Numeric (default = 1). How to deal with knowledge about study protocol. strategy = 1 means select data based on `hrs.del.start` and `hrs.del.end`. strategy = 2 makes that only the data between the first midnight and the last midnight is used. strategy = 3 only selects the most active X days in the file where X is specified by argument `ndayswindow`. strategy = 4 to only use the data after the first midnight.

hrs.del.start Numeric (default = 0). How many HOURS after start of experiment did wearing of monitor start? Used in GGIR g.part2 when `strategy = 1`.

hrs.del.end Numeric (default = 0). How many HOURS before the end of the experiment did wearing of monitor definitely end? Used in GGIR g.part2 when `strategy = 1`.

maxdur Numeric (default = 0). How many DAYS after start of experiment did experiment definitely stop? (set to zero if unknown).

ndayswindow Numeric (default = 7). If `strategy` is set to 3 then this is the size of the window as a number of days.

includedaycrit.part5 Numeric (default = 0.667). Inclusion criteria for number of valid hours, either as expressed as a ratio of 1 or as the number of hours in a 24 hour day.

includedaycrit Numeric (default = 16). 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).

max_calendar_days Numeric (default = 0). The maximum number of calendar days to include (set to zero if unknown).

params_phyact: A list of parameters releated to physical activity as used in GGIR g.part2 and GGIR g.part5.

mvpathreshold Numeric (default = 100). Acceleration threshold for MVPA estimation in GGIR g.part2. This can be a single number or an array of numbers, e.g., `mvpathreshold = c(100, 120)`. In the later case the code will estimate MVPA seperately for each threshold. If this variable is left blank, e.g., `mvpathreshold = c()`, then MVPA is not estimated.

mvpadur Numeric (default = c(1, 5, 10)). The bout duration(s) for which MVPA will be calculated. Only used in GGIR g.part2.

boutcriter Numeric (default = 0.8). A number between 0 and 1, it defines what fraction of a bout needs to be above the mvpathreshold, only used in GGIR g.part2.

threshold.lig Numeric (default = 40). In g.part5: 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 repeated 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 Numeric (default = 100). In g.part5: 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, 120)`. 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 Numeric (default = 400). In g.part5: Threshold for vigorous physical activity to separate moderate from vigorous. Value can be one number or an array of multiple numbers, e.g., `threshold.vig =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.

bout.metric Numeric (default = 6). Specify a metric for bout detection (recommended setting is `bout.metric = 6` since it is the most updated version of the bout calculation in GGIR). If `bout.metric = 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 `bout.metric = 2` the code looks for groups of epochs with a bout.metric 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 may simplify interpretation and still counts the two persons in the example as each others equal. If `bout.metric = 3`, using sliding window across the data to test bout criteria per window and do not allow for breaks larger than 1 minute and with fraction of time larger than the boutcriter threshold. If `bout.metric = 4`, same as 3 but also requires the first and last epoch to meet the threshold criteria. If `bout.metric = 5`, same as 4, but now looks for breaks larger than a minute such that 1 minute breaks are allowe, and the fraction of time that meets the

threshold should be equal than or greater than the `bout.criter` threshold. If `bout.metric = 6`, algorithm improved (2021) to check for first and last epoch.

closedbout Boolean (default = FALSE). TRUE if you want breaks in bouts to be counted towards time spent in bouts (argument only active for bout.metric 1 and 2).

boutdur.mvpa Numeric (default = c(1, 5, 10)). Duration(s) of MVPA bouts in minutes to be extracted. It will start with the identification of the longest to the shortest duration. In the default setting, it will start with the 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 Numeric (default = c(10, 20, 30)). Duration(s) 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. It will start with the identification of the longest to the shortest duration. In the default setting, it 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 Numeric (default = c(1, 5, 10)). Duration(s) 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. It will start with the identification of the longest to the shortest duration. In the default setting, 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.

boutcriter.mvpa Numeric (default = 0.9). A number between 0 and 1, it defines what fraction of a bout needs to be above the `threshold.mod`.

boutcriter.in Numeric (default = 0.9). A number between 0 and 1, it defines what fraction of a bout needs to be below the `threshold.lig`.

boutcriter.lig Numeric (default = 0.8). A number between 0 and 1, it defines what fraction of a bout needs to be between the `threshold.lig` and the `threshold.mod`.

frag.metrics Character (default = NULL). Fragmentation metric to exract. Can be "mean", "TP", "Gini", "power", or "CoV", "NFragPM", or all the above metrics with "all". See package vignette for description of fragmentation metrics.

params_sleep: A list of parameters used to configure the sleep analysis as performend in GGIR g.part3 and g.part4.

relyonguider Boolean (default = FALSE). If TRUE, then sleep onset and waking time are defined based on timestamps derived from the guider. If participants were instructed NOT to wear the accelerometer during waking hours then set to TRUE, in all other scenarios set to FALSE.

relyonsleeplog Boolean (default = NULL). Do not use, now replaced by argument relyonguider. Values provided to argument relyonsleeplog will be passed on to argument relyonguider to not preserve functionality of old R scripts.

def.noc.sleep Numeric (default = 1). 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 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 based on built in algorithms. See argument `HASPT.algo` from HASPT for specifying which of the algorithms to use.

sleepwindowType Character (default = "SPT"). To indicate type of information in the sleeplog, "SPT" for sleep period time. Set to "TimeInBed" if sleep log recorded time in bed to enable calculation of sleep latency and sleep efficiency.

nnights Numeric (default = NULL). 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.

loglocation Character (default = NULL). Path to csv file with sleep log information. See package vignette for how to format this file.

colid Numeric (default = 1). Column number in the sleep log spreadsheet in which the participant ID code is stored.

coln1 Numeric (default = 2). Column number in the sleep log spreadsheet where the onset of the first night starts.

sleeplogidnum Boolean (default = TRUE). Should the participant identifier as stored in the sleeplog be interpretted as a number (TRUE) or character (FALSE)?

ignorenonwear Boolean (default = TRUE). If TRUE then ignore detected monitor non-wear periods to avoid confusion between monitor non-wear time and sustained inactivity.

constrain2range Boolean (default = TRUE). Whether or not to constrain the range of threshold used in the diary free sleep period time window detection.

HASPT.algo Character (default = "HDCZA"). To indicate what algorithm should be used for the sleep period time detection. Default "HDCZA" is Heuristic algorithm looking at Distribution of Change in Z-Angle as described in van Hees et al. 2018. Other options included: "HorAngle", which is based on HDCZA but replaces non-movement detection of the HDCZA algorithm by looking for time segments where the angle of the longitudinal sensor axis has an angle relative to the horizontal plane between -45 and +45 degrees.

HASPT.ignore.invalid Boolean (default = FALSE). To indicate whether invalid time segments should be ignored in the Sleep Period Time detection.

HASIB.algo Character (default = "vanHees2015"). To indicate which algorithm should be used to define the sustained inactivity bouts (i.e., likely sleep). Options: "vanHees2015", "Sadeh1994", "Galland2012".

Sadeh_axis Character (default = "Y"). To indicate which axis to use for the Sadeh1994 algorithm, and other algortihms that relied on count-based Actigraphy such as Galland2012.

sleeplogsep Character (default = ","). Value used as sep argument in read.csv for reading sleeplog csv file, usually "," or ";".

nap_model Character (default = NULL). To specify classification model. Currently the only option is "hip3yr", which corresponds to a model trained with hip data in 3-3.5 olds trained with parent diary data.

longitudinal_axis Integer (default = NULL). To indicate which axis is the longitudinal axis. If not provided, the function will estimate longitudinal axis. Only used when `sensor.location = "hip"` or `HASPT.algo = "HorAngle"`.

anglethreshold Numeric (default = 5). Angle threshold (degrees) for sustained inactivity periods detection. The algorithm will look for periods of time (`timethreshold`) in which the angle variability is lower than `anglethreshold`. This can be specified as multiple thresholds, each of which will be implemented, e.g., `anglethreshold = c(5,10)`.

timethreshold Numeric (default = 5). Time threshold (minutes) for sustained inactivity periods detection. The algorithm will look for periods of time (`timethreshold`) in which the angle variability is lower than `anglethreshold`. This can be specified as multiple thresholds, each of which will be implemented, e.g., `timethreshold = c(5,10)`.

possible_nap_window Numeric (default = c(9, 18)). Numeric vector of length two with range in clock hours during which naps are assumed to take place, e.g., `possible_nap_window = c(9, 18)`.

possible_nap_dur Numeric (default = c(15, 240)). Numeric vector of length two with range in duration (minutes) of a nap, e.g., `possible_nap_dur = c(15, 240)`.

params_247: A list of parameters releated to description of 24/7 behaviours that do not fall under conventional physical activity or sleep outcomes, these parameters are used in GGIR g.part2 and GGIR g.part5:

qwindow Numeric or character (default = c(0, 24)). To specify windows over which all variables are calculated, e.g., acceleration distirbution, number of valid hours, LXMX analysis, MVPA. If numeric, qwindow should have length two, e.g., `qwindow = c(0, 24)`, all variables will only be calculated over the full 24 hours in a day. If `qwindow = c(8, 24)` variables will be calculated over the window 0-8, 8-24 and 0-24. All days in the recording will be segmented based on these values. If you want to use a day specific segmentation in each day then you can set qwindow to be the full path to activity diary file (character). Expected format of the activity diary is: First column headers followed by one row per recording, first column is recording ID, which needs to match with the ID GGIR extracts from the accelerometer file. Followed by date column in format "23-04-2017", where date format is specified by argument qwindow_dateformat (below). Use the character combination date, Date or DATE in the column name. This is followed by one or multiple columns with start times for the activity types in that day format in hours:minutes:seconds. The header of the column will be used as label for each activity type. Insert a new date column before continuing with activity types for next day. Leave missing values empty. If an activitylog is used then individuals who do not appear in the activitylog will still be processed with value `qwindow = c(0, 24)`. Dates with no activiy log data can be skipped, no need to have a column with the date followed by a column with the next date.

qwindow_dateformat Character (default = "%d-%m-%Y"). To specify the date format used in the activity log as used by strptime.

M5L5res Numeric (default = 10). Resolution of L5 and M5 analysis in minutes.

winhr Numeric (default = 5). Vector of window size(s) (unit: hours) of L5 and M5 analysis.

qlevels Numeric (default = NULL). 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.

ilevels Numeric (default = NULL). Levels for acceleration value frequency distribution in m*g*, e.g., `ilevels = c(0,100,200)`. There is no limit to the number of levels. If left empty then the intensity levels will not be extracted. Distribution will be derived from short epoch metric data.

window.summary.size Numeric (default = 10). Functionality designed for the London Centre of Longidutinal studies. Size in minutes of the summary window

iglevels Numeric (default = NULL). Levels for acceleration value frequency distribution in m*g* 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 `iglevels = c(seq(0,4000,by=25), 8000)` or the user could specify their own distribution. There is no constriction to the number of levels.

IVIS_windowsize_minutes Numeric (default = 60). Window size of the Intradaily Variability (IV) and Interdaily Stability (IS) metrics in minutes, needs to be able to add up to 24 hours.

IVIS_epochsize_seconds Numeric (default = NULL). This argument is deprecated.

IVIS.activity.metric Numeric (default = 2). Metric used for activity calculation. Value = 1, uses continuous scaled acceleration. Value = 2, tries to collapse acceleration into a binary score of rest versus active to try to similate the original approach.

IVIS_acc_threshold Numeric (default = 20). Acceleration threshold to distinguish inactive from active.

qM5L5 Numeric (default = NULL). Percentiles (quantiles) to be calculated over L5 and M5 window.

MX.ig.min.dur Numeric (default = 10). Minimum MX duration needed in order for intensity gradient to be calculated.

LUXthresholds Numeric (default = c(0, 100, 500, 1000, 3000, 5000, 10000)). Vector with numeric sequece corresponding to the thresholds used to calculate time spent in LUX ranges.

LUX_cal_constant Numeric (default = NULL). If both LUX_cal_constant and LUX_cal_exponent are provided LUX values are converted based on formula y = constant * exp(x * exponent)

LUX_cal_exponent Numeric (default = NULL). If both LUX_cal_constant and LUX_cal_exponent are provided LUX LUX values are converted based on formula y = constant * exp(x * exponent)

LUX_day_segments Numeric (default = NULL). Vector with hours at which the day should be segmented for the LUX analysis.

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.

cosinor Argument depricated after version 1.5-24. Boolean (default = FALSE). Whether to apply the cosinor analysis from the ActCR package.

params_output: A list of parameters used to specify whether and how GGIR stores its output at various stages of the process.

storefolderstructure Boolean (default = FALSE). Store folder structure of the accelerometer data.

do.part2.pdf Boolean (default = TRUE). In g.part2: Whether to generate a pdf for g.part2.

do.part3.pdf Boolean (default = TRUE). In g.part3: Whether to generate a pdf for g.part3.

timewindow Character (default = c("MM", "WW")). In g.part5: 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").

save_ms5rawlevels Boolean (default = FALSE). In g.part5: Whether to save the time series classification (levels) as csv or RData files (as defined by `save_ms5raw_format`).

save_ms5raw_format Character (default = "csv"). In g.part5: To specify how data should be stored: either "csv" or "RData". Only used if `save_ms5rawlevels = TRUE`.

save_ms5raw_without_invalid Boolean (default = TRUE). In g.part5: To indicate whether to remove invalid days from the time series output files. Only used if `save_ms5rawlevels = TRUE`.

epochvalues2csv Boolean (default = FALSE). In g.part2: If TRUE then epoch values are exported to a csv file. Here, non-wear time is imputed where possible.

do.sibreport Boolean (default = FALSE). In g.part4: To indicate whether to generate report for the sustained inactivity bouts (SIB).

do.visual Boolean (default = TRUE). In g.part4: If TRUE, 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 visually verify that the sleeplog entries do not come with obvious mistakes.

outliers.only Boolean (default = FALSE). In g.part4: Only used if `do.visual = TRUE`. If FALSE, all available nights are included in the visual representation of the data and sleeplog. If TRUE, then only nights with a difference in onset or waking time larger than the variable of argument `criterror` will be included.

criterror Numeric (default = 3). In g.part4: Only used if `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.

visualreport Boolean (default = TRUE). If TRUE, then generate visual report based on combined output from g.part2 and g.part4.

viewingwindow Numeric (default = 1). Centre the day as displayed around noon (`viewingwindow = 1`) or around midnight (`viewingwindow = 2`) in the visual report generated with `visualreport = TRUE`.

week_weekend_aggregate.part5 Boolean (default = FALSE). In g.part5: To indicate whether week and weekend-days aggregates should be stored. This is turned off by default as it generates a large number of extra columns in the output report.

dofirstpage Boolean (default = TRUE). To indicate whether a first page with histograms summarizing the whole measurement should be added in the file summary reports generated with `visualreport = TRUE`.

Value

The function provides no values, it only ensures that other functions are called and that their output is stored. Further, a configuration file is stored containing all the argument values used to facilitate reproducibility.

Author(s)

Vincent T van Hees <v.vanhees@accelting.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(1,2,3,4,5)
  datadir = "C:/myfolder/mydata"
  outputdir = "C:/myresults"
  studyname ="test"
  f0 = 1
  f1 = 2
  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 = 1,
        loglocation= "D:/sleeplog.csv",
        outliers.only = FALSE,
        criterror = 4,
        relyonguider = 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)
```

HASIB *Heuristic algorithms for sustiained inactivty bouts detection*

Description

Apply heuristic algorithms for sustiained inactivty bouts detection. Function not intended for direct use by package user

Usage

```
HASIB(HASIB.algo = "vanHees2015", timethreshold=c(), anglethreshold=c(),
```

```
time=c(), anglez=c(), ws3=c(), zeroCrossingCount=c(), BrondCount = c(),
      NeishabouriCount = c())
```

Arguments

HASIB.algo	Character to indicator which sib algorithm should be used. Default value: "vanHees2015". Other options: "Sadeh1994", "Galland2012", "ColeKripke1992"
anglethreshold	See g.sib.det
timethreshold	See g.sib.det
time	Vector with time per short epoch
anglez	Vector with z-angle per short epoch
ws3	See g.getmeta
zeroCrossingCount	Vector with zero crossing counts per epoch as required for count-based algorithms
BrondCount	Vector with Brond counts per epoch to be used by the count-based algorithms
NeishabouriCount	Vector with Neishabouri counts per epoch to be used by the count-based algorithms

Value

Vector with binary indicator of sustained inactivity bout, 1 is yes, 0 is no.

Author(s)

Vincent T van Hees <v.vanhees@accelting.com>

HASPT *Heuristic Algorithms estimating SPT window.*

Description

As used in function g.sib.det. Function is not intended for direct use by GGIR user.

Usage

```
HASPT(angle, perc = 10, spt_threshold = 15, sptblocksize = 30,
      spt_max_gap = 60, ws3 = 5, constrain2range = FALSE,
      HASPT.algo="HDCZA", invalid, HASPT.ignore.invalid=FALSE)
```

Arguments

angle	Vector of epoch level estimates of angle
perc	Number to indicate percentage threshold (default 10 corresponds to 2018 paper)
spt_threshold	Numeric threshold used in HASPT algorithm (default 15 corresponds to 2018 paper)
sptblocksize	Number to indicate minimum SPT block size (minutes)
spt_max_gap	Number to indicate maximum gap (minutes) in SPT window blocks.
ws3	Number representing epoch length in seconds
constrain2range	Bolean to indicate whether threshold should be constrained to a range
HASPT.algo	Character to indicate what algortihm should be used. Default "HDCZA" is Heuristic algorithm looking at Distribution of Change in Z-Angle as described in van Hees et al. 2018. Other options included: "HorAngle", which is based on HDCZA but replaces non-movement detection of the HDCZA algorithm by looking for time segments where the angle of the longitudinal sensor axis has an angle relative to the horizontal plane between -45 and +45 degrees.
invalid	Integer vector with per epoch an indicator of valid(=0) or invalid(=1) epoch.
HASPT.ignore.invalid	Boolean to indicate whether invalid time segments should be ignored

Value

List with start and end times of the SPT window and the threshold as used.

Author(s)

Vincent T van Hees <v.vanhees@accelting.com>

---

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(ts, TRLi,TRMi,TRVi,
                ws3, params_phyact, ...)
```

Arguments

ts	Data.frame with time series genrated in .gpart5
TRLi	Numeric acceleration threshold light
TRMi	Numeric acceleration threshold moderate
TRVi	Numeric acceleration threshold vigorous
ws3	Numeric size of epoch in seconds
params_phyact	See g.part2
...	Any argument used in the previous version of identify_level, which will now be used to overrule the arguments specified with the parameter objects.

Value

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

Author(s)

Vincent T van Hees <v.vanhees@accelting.com>

Examples

```
  ## Not run:
    levels = identify_levels(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)
```

ismovisens — *Checks whether the files to process are collected with movisens accelerometers.*

Description

Checks whether the files in the datadir folder are files collected with movisens accelerometers. Note that movisens data are stored in one folder per recording that includes multiple bin-files (instead of one file per recording as usual in other accelerometer brands). Therefore, datadir indicates the directory where all the recording folders are stored, then, GGIR reads the pertinent bin files from every folder.

Usage

```
ismovisens(data)
```

Arguments

data	Full path to the recording folder (with the bin files inside) or the datadir (where all the recording folders are stored).

Value

Boolean whether it is a movisens file (TRUE) or not (FALSE)

Examples

```
## Not run:
is.mv = ismovisens(data)

## 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(calendar_date=c(),tz="Europe/London")
```

Arguments

calendar_date	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")
```

---

load_params *Load default parameters*

Description

Loads default paramter values Not intended for direct use by GGIR users.

Usage

```
load_params(group = c("sleep", "metrics", "rawdata", "247",
                      "phyact", "cleaning", "output", "general"))
```

Arguments

group	Character vector with parameter groups to be loaded.

Value

Lists of parameter objects

Author(s)

Vincent T van Hees <v.vanhees@accelting.com>

parametersVignette *Builds Section for Parameters Vignette*

Description

Function extracts the documentation for a given GGIR argument as provided in the GGIR documentation and builds the structure for the Parameters Vignette. Function not designed for direct use by package user.

Usage

```
parametersVignette(params = "sleep")
```

Arguments

params	Character (default = "sleep"). Name of the parameters object to build its corresponding section in the Parameters vignette.

Value

Structure for the vignette subsection.

Author(s)

Jairo Hidalgo Migueles <jairo.hidalgo.migueles@gmail.com>

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 <v.vanhees@accelting.com>

Examples

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

## End(Not run)
```

`read.myacc.csv` *Read custom csv files with accelerometer data*

Description

Loads csv files with accelerometer data and standardises the output format (incl. unit of measurement, timestamp format, header format, and column locations) to make the data compatible with other GGIR functions.

Usage

```
read.myacc.csv(rmc.file=c(), rmc.nrow=Inf, rmc.skip = c(), rmc.dec=".",
                        rmc.firstrow.acc = 1, rmc.firstrow.header=c(),
                        rmc.header.length = c(),
                        rmc.col.acc = 1:3, rmc.col.temp = c(),
                        rmc.col.time=c(),
                        rmc.unit.acc = "g", rmc.unit.temp = "C",
                        rmc.unit.time = "POSIX",
                        rmc.format.time = "%Y-%m-%d %H:%M:%OS",
                        rmc.bitrate = c(), rmc.dynamic_range = c(),
                        rmc.unsignedbit = TRUE,
                        rmc.origin = "1970-01-01",
                        rmc.desiredtz = "Europe/London",
                        rmc.sf = c(),
                        rmc.headername.sf = c(),
                        rmc.headername.sn = c(),
                        rmc.headername.recordingid = c(),
                        rmc.header.structure = c(),
                        rmc.check4timegaps = FALSE,
                        rmc.col.wear = c(),
                        rmc.doresample=FALSE,
                        interpolationType=1,
                        PreviousLastValue = c(0, 0, 1),
                        PreviousLastTime = NULL,
                        epochsize = NULL)
```

Arguments

rmc.file	Filename of file to be read if it is in the working directory, or full path to the file otherwise.
rmc.nrow	Number of rows to read, same as nrow argument in read.csv and nrows in fread. The whole file is read by default (i.e., rmc.nrow = Inf).
rmc.skip	Number of rows to skip, same as skip argument in read.csv and in fread.
rmc.dec	Decimal used for numbers, same as skip argument in read.csv and in fread.
rmc.firstrow.acc	First row (number) of the acceleration data.
rmc.firstrow.header	First row (number) of the header. Leave blank if the file does not have a header.
rmc.header.length	If file has header, specify header length (numeric).
rmc.col.acc	Vector with three column (numbers) in which the acceleration signals are stored
rmc.col.temp	Scalar with column (number) in which the temperature is stored. Leave in default setting if no temperature is avaible. The temperature will be used by g.calibrate.
rmc.col.time	Scalar with column (number) in which the timestamps are stored. Leave in default setting if timestamps are not stored.
rmc.unit.acc	Character with unit of acceleration values: "g", "mg", or "bit"
rmc.unit.temp	Character with unit of temperature values: (K)elvin, (C)elsius, or (F)ahrenheit
rmc.unit.time	Character with unit of timestamps: "POSIX", "UNIXsec" (seconds since origin, see argument rmc.origin), "character", or "ActivPAL" (exotic timestamp format only used in the ActivPAL activity monitor).
rmc.format.time	Character string giving a date-time format as used by strptime. Only used for rmc.unit.time: character and POSIX.
rmc.bitrate	Numeric: If unit of acceleration is a bit then provide bit rate, e.g. 12 bit.
rmc.dynamic_range	Numeric, if unit of acceleration is a bit then provide dynamic range deviation in g from zero, e.g. +/-6g would mean this argument needs to be 6. If you give this argument a character value the code will search the file header for elements with a name equal to the character value and use the corresponding numeric value next to it as dynamic range.
rmc.unsignedbit	Boolean, if unsignedbit = TRUE means that bits are only positive numbers. if unsignedbit = FALSE then bits are both positive and negative.
rmc.origin	Origin of time when unit of time is UNIXsec, e.g. 1970-1-1
rmc.desiredtz	Timezone in which device was configured and experiments took place. If experiments took place in a different timezone, then use this argument for the timezone in which the experiments took place and argument configtz to specify where the device was configured (not implemented yet).

`rmc.sf` Sample rate in Hertz, if this is stored in the file header then that will be used instead.

`rmc.headername.sf`
If file has a header: Row name (character) under which the sample frequency can be found.

`rmc.headername.sn`
If file has a header: Row name (character) under which the serial number can be found.

`rmc.headername.recordingid`
If file has a header: Row name (character) under which the recording ID can be found.

`rmc.header.structure`
Character used to split the header name from the header value, e.g. ":" or " "

`rmc.check4timegaps`
Boolean to indicate whether gaps in time should be imputed with zeros. Some sensing equipment provides accelerometer with gaps in time. The rest of GGIR is not designed for this, by setting this argument to TRUE the the gaps in time will be filled with zeros.

`rmc.col.wear` If external wear detection outcome is stored as part of the data then this can be used by GGIR. This argument specifies the column in which the wear detection (Boolean) is stored.

`rmc.doresample` Boolean to indicate whether to resample the data based on the available timestamps and extracted sample rate from the file header

`interpolationType`
Integer to indicate type of interpolation to be used when resampling time series (mainly relevant for Axivity sensors), 1=linear, 2=nearest neighbour.

`PreviousLastValue`
Automatically identified last value in previous chunk of data read.

`PreviousLastTime`
Automatically identified last timestamp in previous chunk of data read.

`epochsize` Numeric vector of length two, with short and long epoch sizes. Only used by GGIR internally.

Details

To use this function in the context of GGIR use all arguments from this function, except rmc.file, rmc.nrow, and rmc.skip as input for function GGIR or g.part1 and also specify argument rmc.noise, which is not part of this function but needed to tell GGIR what noise level to expect in the data. The rmc.noise is taken from the params_rawdata object if not explicitly specified by user.

Value

List with objects data holding the time series of acceleration, and header if it was available in the orignal file.

Author(s)

Vincent T van Hees <v.vanhees@accelting.com>

Examples

```
# create test files: No header, with temperature, with time
N = 30
sf = 30
timestamps = as.POSIXlt(Sys.time()+((0:(N-1))/sf),origin="1970-1-1",tz = "Europe/London")
mydata = data.frame(x=rnorm(N), time=timestamps,y=rnorm(N),z=rnorm(N),temp=rnorm(N)+20)
testfile = "testcsv1.csv"
write.csv(mydata, file= testfile, row.names = FALSE)
loadedData = read.myacc.csv(rmc.file=testfile, rmc.nrow=20, rmc.dec=".",
                    rmc.firstrow.acc = 1, rmc.firstrow.header=c(),
                    rmc.col.acc = c(1,3,4), rmc.col.temp = 5, rmc.col.time=2,
                   rmc.unit.acc = "g", rmc.unit.temp = "C", rmc.origin = "1970-01-01")
if (file.exists(testfile)) file.remove(testfile)
```

`ShellDoc2Vignette` *Extract Definition from Shell Documentation*

Description

Function extracts the documentation for a given GGIR argument as provided in the GGIR documentation. Function not designed for direct use by package user.

Usage

```
ShellDoc2Vignette(argument = "mode")
```

Arguments

`argument`	Character (default = "mode"). Name of the argument to extract the definition.

Value

Character object with the definition of the argument.

Author(s)

Jairo Hidalgo Migueles <jairo.hidalgo.migueles@gmail.com>

tidyup_df *Round numeric columns and replace NA/NaN values by blank*

Description

Identifies columns that can be coerced to numeric in a data frame, transforms these columns to numeric and round them to the specified digits. It also replaces NA and NaNs values by blank.

Usage

```
tidyup_df(df = c(), digits = 3)
```

Arguments

df	Data frame
digits	Integer indicating the number of decimal places (round) or significant digits (signif) to be used

Value

Data frame with all possible columns as numeric and rounded to the specified number of digits

Author(s)

Jairo H Migueles

Examples

```
# Test data frame
df = data.frame(a = c("a", "b"), b = as.character(c(1.543218, 8.216856483)))
tidyup_df(df = df, digits = 3)
```

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