Package 'GGIR'

May 1, 2020

Type Package

Title Raw Accelerometer Data Analysis

Version 2.0-0

Date 2020-04-30

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

Description A tool to process and analyse data collected with wearable raw acceleration sensors as described in Migueles and colleagues (2019) <doi: 10.1123/jmpb.2018-0063>, van Hees and colleagues (2014) <doi: 10.1152/japplphysiol.00421.2014>, and (2015) <doi: 10.1371/journal.pone.0142533>. The package has been developed and tested for binary data from 'GENEActiv' <https://www.activinsights.com/> and GENEA devices (not for sale), .csv-export data from 'Actigraph' <http://actigraphcorp.com> devices, and .cwa and .wav-format data from 'Axivity' <https://axivity.com>. 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

Imports data.table, Rcpp (>= 0.12.10), foreach, doParallel, signal, zoo, bitops, matlab, GENEAread, tuneR

Depends stats, utils, R (>= 3.5.0)

NeedsCompilation yes

LinkingTo Rcpp

VignetteBuilder knitr

ByteCompile yes

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

Joe Heywood [ctb],
Evgeny Mirkes [ctb],
Severine Sabia [ctb],
Joan Capdevila Pujol [ctb],
Jairo H Migueles [ctb],
Matthew R Patterson [ctb],
Dan Jackson [ctb]

Repository CRAN

Date/Publication 2020-05-01 06:50:28 UTC

R topics documented:

GGIR-package 4
applyExtFunction 5
chartime2iso8601 6
check_myfun 7
createConfigFile 7
create_test_acc_csv 8
create_test_sleeplog_csv 9
data.calibrate 9
data.getmeta 10
data.inspectfile 10
datadir2fnames 11
g.abr.day.names 12
g.analyse 12
g.analyse.avy 15
g.analyse.perday 17
g.analyse.perfile 19
g.applymetrics 21
g.binread 22
g.calibrate 23
g.create.sp.mat 25
g.createcoordinates 25
g.cwaread 26
g.detecmidnight 27
g.dotorcomma 28
g.downsample 29
g.extractheadervars 29
g.getbout 30
g.getidfromheaderobject 31
g.getM5L5 32
g.getmeta 33
g.getstarttime 36
g.impute 37
g.inspectfile 38
g.intensitygradient 39
g.IVIS 40

g.loadlog . . . 41
g.metric . . . 42
g.part1 . . . 43
g.part2 . . . 47
g.part3 . . . 50
g.part4 . . . 52
g.part5 . . . 54
g.part5.addfirstwake . . . 58
g.part5.addsib . . . 59
g.part5.definedays . . . 59
g.part5.fixmissingnight . . . 60
g.part5.onsetwaketiming . . . 61
g.part5.savetimeseries . . . 62
g.part5.wakesleepwindows . . . 62
g.plot . . . 63
g.plot5 . . . 64
g.readaccfile . . . 65
g.report.part2 . . . 67
g.report.part4 . . . 68
g.report.part5 . . . 69
g.shell.GGIR . . . 70
g.sib.det . . . 73
g.sib.plot . . . 75
g.sib.sum . . . 75
g.wavread . . . 76
g.weardec . . . 77
getFirstTimestamp . . . 78
getfolderstructure . . . 78
getStartEnd . . . 79
getStartEndNumeric . . . 80
get_nw_clip_block_params . . . 80
get_starttime_weekday_meantemp_truncdata . . . 81
identify_levels . . . 82
is.ISO8601 . . . 83
isfilelist . . . 84
iso8601chartime2POSIX . . . 84
is_this_a_dst_night . . . 85
numUnpack . . . 85
POSIXtime2iso8601 . . . 86
read.myacc.csv . . . 87
resample . . . 89
updateBlocksize . . . 90

Index **91**

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:	2.0-0
Date:	2020-04-30
License:	LGPL (>= 2.0, < 3)
Discussion group:	https://groups.google.com/forum/#!forum/rpackageggir

Author(s)

- Vincent T van Hees <v.vanhees@movementdata.nl> main developer
- Zhou Fang co-developed function g.calibrate
- Jing Hua Zhao <jinghua.zhao@mrc-epid.cam.ac.uk> co-developed function g.binread
- Joe Heywood helped develop the functionality to process only specific days
- Evgeny Mirkes developed function g.cwaread
- Severine Sabia tested and provided feedback on various functions
- Joan Capdevila Pujol helped to improve various function
- Jairo H Migueles <jairohm@ugr.es> helped to improve various functions
- Dan Jackson helped with unpack function for AX3 data.
- Matthew R Patterson helped with enhancing the visual report.

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 g.shell.GGIR or g.part1. Object `myfun` is a list as detailed below.

Usage

```
applyExtFunction(data, myfun, sf, ws3)
```

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).

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 <vincentvanhees@gmail.com>

---

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 <vincentvanhees@gmail.com>

---

createConfigFile *Creates Config File based on variables in g.shell.GGIR environment*

Description

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

Usage

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

Arguments

config.parameters	List with all arguments used in g.shell.GGIR.

Author(s)

Vincent T van Hees <vincentvanhees@gmail.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())
```

Arguments

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

Value

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

Examples

```
## Not run:
  create_test_sleeplog_csv()

## End(Not run)
```

data.calibrate

Example output from g.calibrate

Description

data.calibrate is example output from g.calibrate

Usage

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

Format

The format is: chr "data.calibrate"

Source

The data was collected on one individual for testing purposes

Examples

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

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

Description

data.getmeta is example output from g.getmeta

Usage

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

Format

The format is: chr "data.getmeta"

Source

The data was collected on one individual for testing purposes

Examples

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

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

Description

data.inspectfile is example output from g.inspectfile

Usage

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

Format

The format is: chr "data.inspectfile"

Source

The data was collected on one individual for testing purposes

Examples

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

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

Description

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

Usage

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

Arguments

`datadir`	See g.part1
`filelist`	Produced by isfilelist

Value

Character vector of filenames

Author(s)

Vincent T van Hees <vincentvanhees@gmail.com>

Examples

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

## End(Not run)
```

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

Description

Abbreviates daynames Monday becomes MON and Sunday becomes SUN

Usage

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

Arguments

daynames	Vector of daynames in character format

Author(s)

Vincent T van Hees <vincentvanhees@gmail.com>

Examples

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

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

Description

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

Usage

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

```
dayborder=0,bout.metric = 1,
closedbout=FALSE,desiredtz = c(),
IVIS_windowsize_minutes = 60,
IVIS_epochsize_seconds = 3600, iglevels = c(),IVIS.activity.metric=1,
qM5L5=c(), myfun=c())
```

Arguments

I	the output from function g.inspectfile
C	the output from function g.calibrate
M	the output from function g.getmeta
IMP	the output from function g.impute
qlevels	array of percentiles for which value needs to be extracted. These need to be expressed as a fraction of 1, e.g. c(0.1, 0.5, 0.75). There is no limit to the number of percentiles. If left empty then percentiles will not be extracted. Distribution will be derived from short epoch metric data, see g.getmeta.
qwindow	To specify windows over which all variables are calculated. If value = c(0,24) all variables will only be calculated over the full 24 hours in a day, If value =c(8,24) variables will be calculated over the window 0-8, 8-24 and 0-24. Previously this functionality was limited to the distribution in acceleration metric values, but now it also derives N valid hours, L5M5 analysis and MVPA.
quantiletype	type of quantile function to use (default recommended). For details, see quantile function in STATS package
L5M5window	Argument depricated after version 1.5-24. This argument used to define the start and end time, in 24 hour clock hours, over which L5M5 needs to be calculated. Now this is done with argument qwindow.
M5L5res	resoltion of L5 and M5 analysis in minutes (default: 10 minutes)
includedaycrit	minimum required number of valid hours in day specific analysis (NOTE: there is no minimum required number of hours per day in the summary of an entire measurement, every available hour is used to make the best possible inference on average metric value per average day)
ilevels	Levels for acceleration value frequency distribution in mg, e.g. c(0,100,200) There is no constriction to the number of levels.
winhr	Vector of window size(s) (unit: hours) of L5 and M5 analysis (dedault = 5 hours)
idloc	If value = 1 (default) the code assumes that ID number is stored in the obvious header field. If value = 2 the code uses the character string preceding the character '_' in the filename as the ID number
snloc	If value = 1 (default) the code assumes that device serial number is stored in the obvious header field. If value = 2 the code uses the character string between the first and second character '_' in the filename as the serial number
mvpathreshold	Threshold for MVPA estimation. This can be a single number or an array of numbers, e.g. c(100,120). In the later case the code will estimate MVPA seperately for each threshold. If this variable is left blank c() then MVPA is not estimated

`boutcriter` The variable boutcriter is a number between 0 and 1 and defines what fraction of a bout needs to be above the mvpathreshold

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

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

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

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

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

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

`desiredtz` see g.getmeta

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

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

`iglevels` Levels for acceleration value frequency distribution in mg used for intensity gradient calculation (according to the method by Rowlands 2018). By default this is argument is empty and the intensity gradient calculation is not done. The user can either provide a single value (any) to make the intensity gradient use the bins c(seq(0,4000,by=25),8000) or the user could specify their own distribution. There is no constriction to the number of levels.

`IVIS.activity.metric` see function g.IVIS

`qM5L5` see function g.getM5L5

`myfun` External function object to be applied to raw data, see g.getmeta.

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 <vincentvanhees@gmail.com>

Examples

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

  #autocalibration:
  C = g.calibrate(datafile)

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

## End(Not run)


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

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

---

g.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(qlevels, doquan, averageday, M, IMP,
  t_TWDI, quantiletype, winhr, L5M5window, M5L5res,
  ws3, IVIS_epochsize_seconds,
  IVIS_windowsize_minutes, IVIS.activity.metric, doiglevels,
  firstmidnighti, ws2, midnightsi, iglevels, qM5L5)
```

Arguments

qlevels	see g.analyse
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
winhr	see g.analyse
L5M5window	see g.analyse
M5L5res	see g.analyse
ws3	Epoch size in seconds
IVIS_epochsize_seconds	see g.IVIS
IVIS_windowsize_minutes	see g.IVIS
IVIS.activity.metric	see g.IVIS
doiglevels	Boolean to indicate whether iglevels should be calculated
firstmidnighti	see g.detecmidnight
ws2	see g.weardec
midnightsi	see g.detecmidnight
iglevels	see g.analyse
qM5L5	see g.getM5L5

Value

InterdailyStability

IntradailyVariability

igfullr_names

igfullr

QUAN

qlevels_names

ML5AD

ML5AD_names

Author(s)

Vincent T van Hees <vincentvanhees@gmail.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(selectdaysfile, ndays, firstmidnighti,
  time, nfeatures, window.summary.size, qwindow, midnightsi,
  metashort, averageday, ENMOi, LFENMOi, BFENi, ENi,
  HFENi, HFENplusi, MADi, ENMOai, doiglevels, nfulldays, lastmidnight,
  ws3, ws2, qcheck, fname, idloc, BodyLocation, wdayname,
  tooshort, includedaycrit, winhr,L5M5window, M5L5res,
  doquan, qlevels, quantiletype, doilevels,
  ilevels, iglevels, domvpa,
  mvpathreshold, boutcriter, closedbout,
  bout.metric, mvpadur, mvpanames, wdaycode, IDd, ID, ID2,
  deviceSerialNumber, qM5L5, ExtFunColsi, myfun)
```

Arguments

selectdaysfile	see g.analyse
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
window.summary.size	see g.analyse
qwindow	see g.analyse
midnightsi	see g.detecmidnight
metashort	see g.impute
averageday	As produced by g.impute
ENMOi	column index of metahosrt where metric is stored
LFENMOi	column index of metahosrt where metric is stored
BFENi	column index of metahosrt where metric is stored
ENi	column index of metahosrt where metric is stored
HFENi	column index of metahosrt where metric is stored
HFENplusi	column index of metahosrt where metric is stored
MADi	column index of metahosrt where metric is stored
ENMOai	column index of metahosrt where metric is stored

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
BodyLocation	as produced by g.extractheadervars
wdayname	character with weekdayname
tooshort	0 (file not too short) or 1 (file too short)
includedaycrit	see g.analyse
winhr	see g.analyse
L5M5window	see g.analyse
M5L5res	see g.analyse
doquan	Boolean whether quantile analysis should be done
qlevels	see g.analyse
quantiletype	see g.analyse
doilevels	Boolean whether to generate ilevels, see g.analyse
ilevels	see g.analyse
iglevels	see g.analyse
domvpa	Boolean whether to do mvpa analysis
mvpathreshold	see g.analyse
boutcriter	see g.analyse
closedbout	see g.analyse
bout.metric	see g.analyse
mvpadur	see g.analyse
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
IDd	As produced by g.extractheadervars
ID	As produced by g.extractheadervars
ID2	gnerated inside g.analyse either a copy of ID, or as character string before the first hyphen
deviceSerialNumber	As produced by g.extractheadervars
qM5L5	see g.getM5L5
ExtFunColsi	column index of metahosrt where metric is stored
myfun	External function object to be applied to raw data, see g.getmeta.

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 <vincentvanhees@gmail.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, ID2, IDd, fname, deviceSerialNumber,
  BodyLocation, 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, IVIS_epochsize_seconds, qwindow)
```

Arguments

ID	see g.analyse.perday
ID2	see g.analyse.perday
IDd	see g.analyse.perday
fname	see g.analyse.perday
deviceSerialNumber	As produced by g.extractheadervars
BodyLocation	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
IVIS_epochsize_seconds	see g.IVIS
qwindow	see g.analyse

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 <vincentvanhees@gmail.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,n,sf,ws3,metrics2do,lb=0.2, hb=15)
```

Arguments

data	Three column matrix with x, y, and z acceleration data
n	filter order, only needed if a metric is selected that involves a frequency filter
sf	sample frequency
ws3	Epoch size in seconds
metrics2do	Dataframe with Boolean indicator for all metrics whether they should be extracted or not. For instance, metrics2do$do.bfen = TRUE, indicates that the bfen metric should be extracted
lb	see g.metric
hb	see g.metric

Value

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

Author(s)

Vincent T van Hees <vincentvanhees@gmail.com>

Examples

```
Gx = runif(n=10000,min=0,max=2)
Gy = runif(n=10000,min=1,max=3)
Gz = runif(n=10000,min=0,max=2)
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)
extractedmetrics = g.applymetrics(data,n=4,sf=40,ws3=5,metrics2do)
```

---

g.binread

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

Description

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

Usage

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

Arguments

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

Details

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

Value

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

Author(s)

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

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

Description

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

Usage

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

Arguments

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

Value

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

Author(s)

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

References

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

Examples

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

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

## End(Not run)
```

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

Description

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

Usage

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

Arguments

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

Value

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

Author(s)

Vincent T van Hees <vincentvanhees@gmail.com>

`g.createcoordinates` *Create coordinates for g.plot*

Description

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

Usage

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

Arguments

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

Value

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

Author(s)

Vincent van Hees <vincentvanhees@gmail.com>

---

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

Description

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

Usage

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

Arguments

fileName	filename (required)
start	start point for reading data, this can either be a timestamp "year-month-day hr:min:sec" or a page number (optional)
end	end point for reading data, this can either be a timestamp "year-month-day hr:min:sec" or a page number (optional)
progressBar	Is trigger to switch on/off the text progress bar. If progressBar is TRUE then the function displays the progress bar but it works slightly slower
desiredtz	Desired timezone, see documentation g.getmeta
configtz	Only functional for AX3 cwa 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.

Value

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

Author(s)

Evgeny Mirkes <em322@leicester.ac.uk>

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

Description

Detect all midnights in a time series

Usage

```
g.detecmidnight(time,desiredtz, 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 <vincentvanhees@gmail.com>

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

Description

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

Usage

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

Arguments

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

Examples

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

## End(Not run)
```

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

Description

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

Usage

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

Arguments

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

Value

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

Author(s)

Vincent T van Hees <vincentvanhees@gmail.com>

Examples

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

g.extractheadervars *Extracts header variables from header object*

Description

Function is not intended for direct interaction by package end user

Usage

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

Arguments

I	Object produced by g.inspectfile

Value

- ID = participant identifier
- iid = investigator identifier
- HN = handedness
- BodyLocation = Attachement location of the sensor
- SX = sex
- deviceSerialNumber = serial number

Author(s)

Vincent T van Hees <vincentvanhees@gmail.com>

Examples

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

---

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

Description

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

Usage

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

Arguments

x	vector of zeros and/or ones to be screened for bouts of ones
boutduration	duration of bout in epochs
boutcriter	Minimum percentage of boutduration for which the epoch values are expected to meet the threshold criterium
closedbout	TRUE if you want breaks in bouts to be counted towards time spent in bouts (argument only active for bout.metric 1 and 2)

bout.metric — If value=1 the code uses the MVPA bout definition as has been available since 2014 (see papers by Sabia AJE 2014 and da Silva IJE 2014). Here, the algorithm looks for 10 minute windows in which more than XX percent of the epochs are above mvpathreshold, and then counts the entire window as mvpa. If value=2 the code looks for a group or groups of epochs with a value above mvpathreshold that span a time window of at least mvpadur minutes in which more than boutcriter percent of the epochs are above the threshold. The motivation for the defition 1 was: A person who spends 10 minutes in MVPA with a 2 minute break in the middle is equally active as a person who spends 8 minutes in MVPA without taking a break. Therefore, both should be counted equal and counted as 10 minute MVPA bout. The motivation for the definition 2 is: not counting breaks towards MVPA simplifies interpretation and still counts the two persons in the example as each others equal. If value=3, using sliding window across the data to test bout criteria per window and do not allow for breaks of 1 minute or longer. If value=4, same as 3 but also requires the first and last epoch to require the threshold criteria.

ws3 — epoch length in seconds, only needed for bout.metric =3, because it needs to measure how many epochs equal 1 minute breaks

Value

x — Vector with binary numbers indicator where bouts where detected

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

Author(s)

Vincent T van Hees <vincentvanhees@gmail.com>

Examples

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

---

g.getidfromheaderobject

Extracts participant identifier from header object

Description

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

Usage

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

Arguments

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

Value

Participant identifier as character

Author(s)

Vincent T van Hees <vincentvanhees@gmail.com>

Examples

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

## End(Not run)
```

---

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

Description

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

Usage

```
g.getM5L5(varnum,ws3,t0_LFMF,t1_LFMF,M5L5res,winhr,qM5L5=c())
```

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.

Value

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

Author(s)

Vincent T van Hees <vincentvanhees@gmail.com>

Examples

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

---

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

Description

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

Usage

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

Arguments

datafile	name of accelerometer file
desiredtz	desired timezone: see also http://en.wikipedia.org/wiki/Zone.tab
windowsizes	Three values to indicate the lengths of the windows as in c(window1,window2,window3): window1 is the short epoch length in seconds and by default 5 this is the time window over which acceleration and angle metrics are calculated, window2 is the long epoch length in seconds for which non-wear and signal clipping are defined, default 900. However, window3 is the window length of data used for non-wear detection and by default 3600 seconds. So, when window3 is larger than window2 we use overlapping windows, while if window2 equals window3 non-wear periods are assessed by non-overlapping windows. Window2 is expected to be a multitude of 60 seconds.
daylimit	number of days to limit (roughly), if set to FALSE no daylimit will be applied
offset	offset correction value per axis, usage: value = scale(value,center = -offset, scale = 1/scale)
scale	scaling correction value per axis, usage: value = scale(value,center = -offset, scale = 1/scale)
tempoffset	temperature offset correction value per axis, usage: value = scale(value,center = -offset, scale = 1/scale) + scale(temperature, center = rep(averagetemperate,3), scale = 1/tempoffset)
do.bfen	if TRUE, calculate metric BFEN with band-pass filter configuration set by lb and hb
do.enmo	if TRUE (default), calculate metric ENMO with negative values rounded to zero
do.lfenmo	if TRUE, calculate metric LFENMO with low-pass filter configuration set by hb
do.en	if TRUE, calculate metric EN
do.hfen	if TRUE, calculate metric HFEN with low-pass filter configuration set by hb
do.hfenplus	if TRUE, calculate metric HFENplus with band-pass filter configuration set by lb and hb
do.mad	if TRUE, calculate metric MAD (Mean Amplitude Deviarion)
do.anglex	if TRUE, calculate the angle of the x-axis relative to the horizontal plane (degrees) utilizing all three axes
do.angley	if TRUE, calculate the angle of the y-axis relative to the horizontal plane (degrees) utilizing all three axes
do.anglez	if TRUE, calculate the angle of the z-axis relative to the horizontal plane (degrees) utilizing all three axes
do.enmoa	if TRUE (default), calculate metric ENMOa which is equal to metric ENMO but with the absolute taken from the Euclidean norm minus one.
do.roll_med_acc_x	if TRUE, calculate rolling median for the x axis
do.roll_med_acc_y	if TRUE, calculate rolling median for the y axis
do.roll_med_acc_z	if TRUE, calculate rolling median for the z axis

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

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

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

`do.lfen`
: if TRUE, calculate metric EN with low-pass filtered inputed signals using configuration set by `hb`

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

`hb`
: upper boundary of the frequency filter (in Hertz)

`n`
: order of the frequency filter

`meantempcal`
: mean temperature corresponding to the data as used for autocalibration. If autocalibration is not done or if temperature was not available then leave blank (default)

`chunksize`
: number between 0.2 and 1 to specificy the size of chunks to be loaded as a fraction of a 24 hour period, e.g. 0.5 equals 12 hour chunks. The default is 1 (24 hrs). For machines with less than 4 Gb of RAM memory a value below 1 is recommended.

`selectdaysfile`
: see g.part1

`dayborder`
: see g.part1

`dynrange`
: see g.part1

`configtz`
: Only functional for AX3 cwa 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.

`myfun`
: External function object to be applied to raw data. See details applyExtFunction.

`...`
: Please ignore. Only used by the code internally when called from within g.part1 with selectdaysfile specific.

Value

`metalong`
: dataframe with long epoch meta-data: EN, non-wear score, clipping score, temperature

`metashort`
: dataframe with short epoch meta-data: timestamp and metric

`tooshort`
: indicator of whether file was too short for processing (TRUE or FALSE)

`corrupt`
: indicator of whether file was considered corrupt (TRUE or FALSE)

Author(s)

Vincent T van Hees <vincentvanhees@gmail.com>

References

- van Hees VT, Gorzelniak L, Dean Leon EC, Eder M, Pias M, et al. (2013) Separating Movement and Gravity Components in an Acceleration Signal and Implications for the Assessment of Human Daily Physical Activity. PLoS ONE 8(4): e61691. doi:10.1371/journal.pone.0061691
- Aittasalo M, Vaha-Ypya H, Vasankari T, Husu P, Jussila AM, and Sievanen H. Mean amplitude deviation calculated from raw acceleration data: a novel method for classifying the intensity of adolescents physical activity irrespective of accelerometer brand. BMC Sports Science, Medicine and Rehabilitation (2015).

Examples

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

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

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

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

## End(Not run)
```

---

g.getstarttime *Extract start time of a measurement*

Description

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

Usage

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

Arguments

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

Value

The starttime

Author(s)

Vincent T van Hees <vincentvanhees@gmail.com>

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

Description

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

Usage

```
g.impute(M, I, strategy = 1, hrs.del.start = 0, hrs.del.end = 0,
maxdur = 0, ndayswindow = 7,desiredtz="", dayborder= 0, TimeSegments2Zero =c())
```

Arguments

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

Value

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

Author(s)

Vincent T van Hees <vincentvanhees@gmail.com>

Examples

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

  #autocalibration:
  C = g.calibrate(datafile)

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

## End(Not run)

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

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

---

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

Description

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

Usage

```
g.inspectfile(datafile, desiredtz = "", ...)
```

Arguments

datafile	name of data file
desiredtz	Desired timezone, see documentation g.getmeta
...	Any input arguments needed for function read.myacc.csv if you are working with a non-standard csv formatted files.

Value

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

Author(s)

Vincent T van Hees <vincentvanhees@gmail.com>

---

g.intensitygradient *Intensity gradient calculation*

Description

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

Usage

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

Arguments

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

Value

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

Author(s)

Vincent T van Hees <vincentvanhees@gmail.com>

References

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

---

g.IVIS *Calculates IV and IS*

Description

To extract interdaily stability and interdaily variability as originally proposed by van Someren. Note that we had to improvise our replication of his metric, because GGIR does not attempt to calculate count values. The implementation in this function has so far not be evaluated yet.

Usage

```
g.IVIS(Xi, epochsizesecondsXi = 5, IVIS_epochsize_seconds = 3600,
IVIS_windowsize_minutes = 60, IVIS.activity.metric = 1)
```

Arguments

`Xi`	Vector with acceleration values, e.g. ENMO metric.
`epochsizesecondsXi`	Epoch size of the values in Xi expressed in seconds.
`IVIS_epochsize_seconds`	Epoch size of the Intradaily Variability (IV) and Interdaily Stability (IS) metrics in seconds
`IVIS_windowsize_minutes`	Window size of the Intradaily Variability (IV) and Interdaily Stability (IS) metrics in minutes
`IVIS.activity.metric`	Metric used for activity calculation. Value = 1, uses continuous scaled acceleration. Value = 2, tries to collapse itinto a binary score of rest versus active to try to similate the original approach.

Value

`InterdailyStability`

`IntradailyVariability`

Author(s)

Vincent T van Hees <vincentvanhees@gmail.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)
```

Arguments

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

Value

Data frame with sleeplog

Author(s)

Vincent T van Hees <vincentvanhees@gmail.com>

Examples

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

## End(Not run)
```

`g.metric` *Extract metrics from acceleration signals*

Description

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

Usage

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

Arguments

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

Value

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

Author(s)

Vincent T van Hees <vincentvanhees@gmail.com>

Examples

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

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

Description

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

Usage

```
g.part1(datadir=c(),outputdir=c(),f0=1,f1=c(),
        windowsizes = c(5,900,3600),
        desiredtz = "",chunksize=c(),studyname=c(),
        do.enmo = TRUE,do.lfenmo = FALSE,do.en = FALSE,
        do.bfen = FALSE, do.hfen=FALSE, do.hfenplus = FALSE,
        do.mad = FALSE, do.anglex=FALSE, do.angley=FALSE,
        do.anglez=FALSE, do.enmoa=FALSE,
        do.roll_med_acc_x=FALSE, do.roll_med_acc_y=FALSE,
        do.roll_med_acc_z=FALSE, do.dev_roll_med_acc_x=FALSE,
        do.dev_roll_med_acc_y=FALSE, do.dev_roll_med_acc_z=FALSE,
        do.lfen = FALSE,
        do.cal = TRUE,lb = 0.2, hb = 15, n = 4,
        spherecrit=0.3,minloadcrit=72,
        printsummary=TRUE,print.filename=FALSE,overwrite=FALSE,
        backup.cal.coef="retrieve",selectdaysfile=c(),dayborder=0,
        dynrange=c(), configtz=c(), do.parallel=TRUE, minimumFileSizeMB = 2,
        myfun=c(), ...)
```

Arguments

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

desiredtz — see g.getmeta

chunksize — see g.getmeta

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

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

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

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

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

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

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

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

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

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

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

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

do.roll_med_acc_x — see g.getmeta

do.roll_med_acc_y — see g.getmeta

do.roll_med_acc_z — see g.getmeta

do.dev_roll_med_acc_x — see g.getmeta

do.dev_roll_med_acc_y — see g.getmeta

do.dev_roll_med_acc_z — see g.getmeta

do.lfen — see g.getmeta

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

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

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

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

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

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

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

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

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

`backup.cal.coef`
Default "retrieve". Option to use backed-up calibration coefficient instead of deriving the calibration coefficients when analysing the same file twice, see details.

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

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

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

`configtz` Only functional for AX3 cwa 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.

`do.parallel` Boolean whether to use multi-core processing (only works if at least 4 CPU cores are available.

`minimumFileSizeMB`
Minimum File size in MB required to enter processing, default 2MB. 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 short files.

`myfun` External function object to be applied to raw data. See details applyExtFunction.

`...` Any input arguments needed for function read.myacc.csv if you are working with a non-standard csv formatted files.

Details

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".

Value

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

Author(s)

Vincent T van Hees <vincentvanhees@gmail.com>

References

- van Hees VT, Gorzelniak L, Dean Leon EC, Eder M, Pias M, et al. (2013) Separating Movement and Gravity Components in an Acceleration Signal and Implications for the Assessment of Human Daily Physical Activity. PLoS ONE 8(4): e61691. doi:10.1371/journal.pone.0061691
- van Hees VT, Fang Z, Langford J, Assah F, Mohammad A, da Silva IC, Trenell MI, White T, Wareham NJ, Brage S. Auto-calibration of accelerometer data for free-living physical activity assessment using local gravity and temperature: an evaluation on four continents. J Appl Physiol (1985). 2014 Aug 7
- Aittasalo M, Vaha-Ypya H, Vasankari T, Husu P, Jussila AM, and Sievanen H. Mean amplitude deviation calculated from raw acceleration data: a novel method for classifying the intensity of adolescents physical activity irrespective of accelerometer brand. BMC Sports Science, Medicine and Rehabilitation (2015).

Examples

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

## End(Not run)
```

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

Description

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

Usage

```
g.part2(datadir=c(),metadatadir=c(),f0=c(),f1=c(),strategy = 1,
hrs.del.start = 0.5,hrs.del.end = 0.5, maxdur = 7,
includedaycrit = 16, L5M5window = c(0,24), M5L5res = 10,
winhr = 5, qwindow=c(0,24), qlevels = c(0.1),
ilevels = c(0,10), mvpathreshold = c(100),
boutcriter = 0.8,ndayswindow=7,idloc=1,
do.imp=TRUE,storefolderstructure=FALSE,overwrite=FALSE,
epochvalues2csv=FALSE, mvpadur=c(1,5,10),selectdaysfile=c(),
window.summary.size=10,dayborder=0,
bout.metric=2,closedbout=FALSE,desiredtz="",
IVIS_windowsize_minutes = 60, IVIS_epochsize_seconds = 3600,
iglevels = c(), IVIS.activity.metric=1, TimeSegments2ZeroFile = c(),
qM5L5=c(), do.parallel = TRUE, myfun=c())
```

Arguments

datadir	Directory where the accelerometer files are stored or list, e.g. "C:/mydata" of accelerometer filenames and directories, e.g. c("C:/mydata/myfile1.bin", "C:/mydata/myfile2.bin").
metadatadir	Directory where the output from g.part1 was stored
f0	File index to start with (default = 1). Index refers to the filenames sorted in increasing order
f1	File index to finish with (defaults to number of files available)
strategy	how to deal with knowledge about study protocol. value = 1 to select data based on hrs.del.start, hrs.del.end, and maxdur. Value = 2 to only use the data between the first midnight and the last midnight, value = 3 only selects the most active X days in the files. X is specified by argument ndayswindow See also g.impute
hrs.del.start	how many HOURS after start of experiment did wearing of monitor start?, see g.impute
hrs.del.end	how many HOURS before the end of the experiment did wearing of monitor definitely end?, see g.impute
maxdur	how many DAYS after start of experiment did experiment definitely stop? (set to zero if unknown = default), see g.impute

includedaycrit minimum required number of valid hours in day specific analysis (NOTE: there is no minimum required number of hours per day in the summary of an entire measurement, every available hour is used to make the best possible inference on average metric value per week)

L5M5window Argument depricated after version 1.5-24. This argument used to define the start and end time, in 24 hour clock hours, over which L5M5 needs to be calculated. Now this is done with argument qwindow.

M5L5res resoltion of L5 and M5 analysis in minutes (default: 10 minutes)

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

qwindow see g.analyse

qlevels array of percentiles for which value needs to be extracted. These need to be expressed as a fraction of 1, e.g. c(0.1, 0.5, 0.75). There is no limit to the number of percentiles. If left empty then percentiles will not be extracted. Distribution will be derived from short epoch metric data, see g.getmeta.

ilevels Levels for acceleration value frequency distribution in mg, e.g. c(0,100,200) There is no constriction to the number of levels.

mvpathreshold Threshold for MVPA estimation. Threshold needs to be based on metric ENMO. This can be a single number or an array of numbers, e.g. c(100,120). In the later case the code will estimate MVPA seperately for each threshold. If this variable is left blank c() then MVPA is not estimated

boutcriter The variable boutcriter is a number between 0 and 1 and defines what fraction of a bout needs to be above the mvpathreshold

ndayswindow If strategy is set to 3 then this is the size of the window as a number of days

idloc If value = 1 (default) the code assumes that ID number is stored in the obvious header field. If value = 2 the code uses the character string preceding the character '_' in the filename as the ID number

do.imp Whether to impute missing values (e.g. suspected of monitor non-wear) or not by g.impute. Default and recommended setting is TRUE

storefolderstructure
Store folder structure of the accelerometer data

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

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

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

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

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

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

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

closedbout — See g.getbout

desiredtz — see g.getmeta

IVIS_windowsize_minutes — see function g.IVIS

IVIS_epochsize_seconds — see function g.IVIS

iglevels — see function g.analyse

TimeSegments2ZeroFile — Csv-file holding the data.frame used for argument TimeSegments2Zero in function g.impute

IVIS.activity.metric — see function g.IVIS

qM5L5 — see function g.getM5L5

do.parallel — Boolean whether to use multi-core processing (only works if at least 4 CPU cores are available.

myfun — External function object to be applied to raw data, see g.getmeta.

Value

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

Author(s)

Vincent T van Hees <vincentvanhees@gmail.com>

References

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

Examples

```
## Not run:
metadatadir = "C:/myresults/output_mystudy"
g.part2(metadatadir)

## End(Not run)
```

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

Description

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

Usage

```
g.part3(metadatadir=c(),f0,f1,anglethreshold = 5,timethreshold = 5,
  acc.metric="ENMO", ignorenonwear=TRUE, overwrite=FALSE,
  desiredtz="",constrain2range=TRUE,
  do.part3.pdf=TRUE, do.parallel=TRUE, dayborder=0, myfun=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 g.shell.GGIR will recognise what the value of metadatadir is.
f0	File index to start with (default = 1). Index refers to the filenames sorted in increasing order
f1	File index to finish with (defaults to number of files available)
anglethreshold	Angle threshold (degrees) for sustained inactivity periods detection, default = 5
timethreshold	Time threshold (minutes) for sustained inactivity periods detection, default = 5. This can be specified as multiple thresholds, each of which will be implemented. For example, timethreshold = c(5,10)
acc.metric	Which one of the metrics do you want to consider to analyze L5. The metric of interest need to be calculated in M (see g.part1)
ignorenonwear	If TRUE then ignore detected monitor non-wear periods to avoid confusion between monitor non-wear time and sustained inactivity (default = TRUE)
overwrite	Overwrite previously generated milestone data by this function for this particular dataset? If FALSE then it will skip the previously processed files (default = FALSE).
desiredtz	See g.getmeta
constrain2range	Whether or not to constrain the range of threshold used in the diary free Sleep period time window detection
do.part3.pdf	Whether to generate a pdf for part 3 (default is TRUE). Turning this off could speed up the processing.

do.parallel	Boolean whether to use multi-core processing (only works if at least 4 CPU cores are available.
dayborder	See g.part1
myfun	External function object to be applied to raw data. See details applyExtFunction.

Value

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

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

Author(s)

Vincent T van Hees <vincentvanhees@gmail.com>

References

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

Examples

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

## End(Not run)
```

g.part4 *Labels detected sustained inactivity periods by g.part3 as either 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,idloc=1,
loglocation = c(),colid = 1,coln1 = 9,nnights = 7,
sleeplogidnum=FALSE,do.visual=FALSE,outliers.only = FALSE,
    excludefirstlast=FALSE,criterror = 1,includenightcrit=16,
    relyonguider=FALSE, relyonsleeplog=FALSE, def.noc.sleep=1,
    storefolderstructure=FALSE,overwrite=FALSE,
    desiredtz="",data_cleaning_file=c())
```

Arguments

datadir	Directory where the accelerometer files are stored or list of accelerometer filenames and directories
metadatadir	Directory that holds a folders 'meta' and inside this a folder 'basic' which contains the milestone data produced by g.part1. The folderstructure is normally created by g.part3. When using g.part4 via g.shell.GGIR then g.shell.GGIR will automatically recognise what the value of metadatadir is, so the user does not need to specify this.
f0	File index to start with (default = 1). Index refers to the filenames sorted in increasing order
f1	File index to finish with (defaults to number of files available)
idloc	If value = 1 (default) the code assumes that ID number is stored in the obvious header field. If value = 2 the code uses the character string preceding the character '_' in the filename as the ID number
loglocation	Location of the spreadsheet (csv) with sleep log information. The spreadsheet needs to have the following structure: one column for participant ID, and then followed by alternatingly one column for onset time and one column for waking time. There can be multiple sleeplogs in the same spreadsheet. The first raw of the spreadsheet needs to be filled with column names, it does not matter what these column names are. Timestamps are to be stored without date as in hh:mm:ss. If onset corresponds to lights out or intention to fall asleep, then it is the end-users responsibility to account for this in the interpretation of the results.
colid	Column number in the sleep log spreadsheet in which the participant ID code is stored (default = 1)

coln1 Column number in the sleep log spreadsheet where the onset of the first night starts

nnights Number of nights for which sleep log information should be available. It assumes that this is constant within a study. If sleep log information is missing for certain nights then leave these blank

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

do.visual If g.part4 is run with do.visual == TRUE then the function will generate a pdf with a visual representation of the overlap between the sleeplog entries and the accelerometer detections. This can be used to visualy verify that the sleeplog entries do not come with obvious mistakes.

outliers.only Relevant for do.visual == TRUE. Outliers.only == FALSE will visualise all available nights in the data. Outliers.only == TRUE will visualise only for nights with a difference in onset or waking time larger than the variable of argument criterror.

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

criterror Relevant for do.visual == TRUE and outliers.only == TRUE. criterror specifies the number of minimum number of hours difference between sleep log and accelerometer estimate for the night to be included in the visualisation

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

relyonguider 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 (default).

relyonsleeplog Now replaced by relyonguider. Values provided to argument relyonsleeplog will be passed on to argument relyonguider to not preserve functionality of old R script.

def.noc.sleep The time window during which sustained inactivity will be assumed to represent sleep, e.g. def.noc.sleep=c(21,9). This is only used if no sleep log entry is available. If def.noc.sleep is left blank 'def.noc.sleep=c()' then the 12 hour window centred at the least active 5 hours of the 24 hour period will be used instead. Here, L5 is hardcoded and will not change by changing argument winhr in function g.part2. If def.noc.sleep is filled with a single integer, e.g. def.noc.sleep=c(1) then the window will be detected with the method as described in van Hees et al. 2018 Scientific Reports.

storefolderstructure
Store folder structure of the accelerometer data

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

desiredtz See g.getmeta

data_cleaning_file
: Optional path to a csv file you create as used that holds three columns: ID, day_part5, and relyonguider_part4 This to facilitate forced exclusion of specific days in the output and/or forced relying on guider in part 4 output. Columns day_part5 and relyonguider_part4 do not both have to have a value.

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 <vincentvanhees@gmail.com>

References

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

Examples

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

## End(Not run)
```

---

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

Description

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

Usage

```
g.part5(datadir=c(), metadatadir=c(), f0=c(), f1=c(), strategy=1,
  maxdur=7, hrs.del.start=0, hrs.del.end =0,
  loglocation= c(), excludefirstlast.part5=FALSE,
  windowsizes=c(5,900,3600),acc.metric="ENMO", boutcriter.mvpa=0.8,
  boutcriter.in=0.9, boutcriter.lig=0.8, storefolderstructure=FALSE,
  threshold.lig = c(40), threshold.mod = c(100),
  threshold.vig = c(400), timewindow=c("MM","WW"),
  boutdur.mvpa = c(1,5,10), boutdur.in = c(10,20,30),
  boutdur.lig = c(1,5,10), winhr = 5, M5L5res = 10,
  overwrite=FALSE, desiredtz="",
  bout.metric=4, dayborder=0, save_ms5rawlevels=FALSE, do.parallel=TRUE,
  part5_agg2_60seconds = FALSE, save_ms5raw_format="csv",
  save_ms5raw_without_invalid=TRUE)
```

Arguments

`datadir`	Directory where the accelerometer files are stored or list of accelerometer filenames and directories
`metadatadir`	Directory that holds a folders 'meta' and inside this a folder 'basic' which contains the milestone data produced by g.part1. The folderstructure is normally created by g.part1 and g.shell.GGIR will recognise what the value of metadatadir is.
`f0`	File index to start with (default = 1). Index refers to the filenames sorted in increasing order
`f1`	File index to finish with (defaults to number of files available)
`strategy`	how to deal with knowledge about study protocol. value = 1 means select data based on `hrs.del.start`, `hrs.del.end`, and `maxdur`. Value = 2 makes that only the data between the first midnight and the last midnight is used for imputation, see g.impute
`maxdur`	how many DAYS after start of experiment did experiment definitely stop? (set to zero if unknown = default), see g.impute
`hrs.del.start`	how many HOURS after start of experiment did wearing of monitor start?, see g.impute
`hrs.del.end`	how many HOURS before the end of the experiment did wearing of monitor definitely end?, see g.impute
`loglocation`	Location of the spreadsheet (csv) with sleep log information. The spreadsheet needs to have the following structure: one column for participant id, and then followed by alternatingly one column for onset time and one column for waking time. Timestamps are to be stored without date as in 18:20:00. If onset corresponds to lights out or intention to fall asleep, then it is the end-users responsibility to account for this in the interpretation of the results.
`excludefirstlast.part5`	If TRUE then the first and last night of the measurement are ignored for the sleep assessment.

`windowsizes` see g.getmeta

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

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

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

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

`storefolderstructure` Store folder structure of the accelerometer data

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

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

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

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

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

`boutdur.in` Durations of inactivty bouts in minutes to be extracted. Inactivity bouts are detected in the segments of the data which were not labelled as sleep or MVPA bouts. The default duration values is c(10,20,30), this will start with the identification of 30 minute bouts, followed by 20 minute bouts in the rest of the data, and followed by 10 minute bouts in the rest of the data.

`boutdur.lig` Durations of light activty bouts in minutes to be extracted. Light activity bouts are detected in the segments of the data which were not labelled as sleep, MVPA, or inactivity bouts. The default duration values is c(1,5,10), this will start with the identification of 10 minute bouts, followed by 5 minute bouts in the rest of the data, and followed by 1 minute bouts in the rest of the data.

M5L5res	resoltion of L5 and M5 analysis in minutes (default: 10 minutes)
overwrite	Overwrite previously generated milestone data by this function for this particular dataset. If FALSE then it will skip the previously processed files (default = FALSE).
desiredtz	see g.getmeta
bout.metric	See documnetion in g.getbout and
dayborder	Hour at which days start and end (default = 0), value = 4 would mean 4am
winhr	see g.getmeta
save_ms5rawlevels	boolean, whether to save the time series classification (levels) as a csv files
do.parallel	Boolean whether to use multi-core processing (only works if at least 4 CPU cores are available.
part5_agg2_60seconds	Boolean whether to use aggregate epochs to 60 seconds as part of the part 5 analysis.
save_ms5raw_format	Character string to specify how data should be stored: either "csv" (default) or "RData". Only used if save_ms5rawlevels is set to TRUE.
save_ms5raw_without_invalid	Boolean to indicate whether to remove invalid days from the time series output files. Only used if save_ms5rawlevels is set to TRUE.

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 <vincentvanhees@gmail.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, sptwindow_HDCZA_end, ws3new)
```

Arguments

`ts`

`summarysleep_tmp2`

`nightsi`

`sleeplog`

`ID`

`Nepochsinhour`

`Nts`

`sptwindow_HDCZA_end`

`ws3new`

Value

Data.frame ts

Author(s)

Vincent T van Hees <vincentvanhees@gmail.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 <vincentvanhees@gmail.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, summarysleep_tmp2, indjump, nightsi_bu,
                        ws3new, qqq_backup=c(), ts, Nts, timewindowi, Nwindows)
```

Arguments

nightsi

wi

summarysleep_tmp2

indjump

nightsi_bu

ws3new

qqq_backup

ts

Nts

timewindowi

Nwindows

Value

List of qqq and qqq_backup

Author(s)

Vincent T van Hees <vincentvanhees@gmail.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 <vincentvanhees@gmail.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 <vincentvanhees@gmail.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)
```

Arguments

ts	
LEVELS	
desiredtz	
rawlevels_fname	
save_ms5raw_format	See g.part5
save_ms5raw_without_invalid	See g.part5

Value

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

Author(s)

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

Examples

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

  #autocalibration:
  C = g.calibrate(datafile)

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

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

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

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

`g.plot5` *Generate user-friendly visual report. 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 g.shell.GGIR to generate report. Not intended for direct use by user

Usage

```
g.plot5(metadatadir = c(), dofirstpage = FALSE, viewingwindow = 1,
f0 = c(), f1 = c(), overwrite = FALSE, metric="ENMO",desiredtz = "Europe/London",
 threshold.lig = 30, threshold.mod = 100, threshold.vig = 400)
```

Arguments

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

Usage

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

Arguments

filename	filename
blocksize	Size of blocks (in file pages) to be read
blocknumber	Block number relative to start of file
selectdaysfile	See documentation g.getmeta
filequality	Single row dataframe with columns: filetooshort, filecorrupt, and filedoesnotholdday. All with the value TRUE or FALSE
decn	Character with a dot or a comma, used for interpretting samplefrequency in the file header. decn is derived with g.dotorcomma
dayborder	See documentation g.part1
ws	Larger windowsize for non-detection, see documentation g.part2
desiredtz	Desired timezone, see documentation g.getmeta
PreviousEndPage	Page number on which previous block ended (automatically assigned within g.getmeta and g.calibrate).
inspectfileobject	Output from the function g.inspectfile.
configtz	Only functional for AX3 cwa 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.
...	Any input arguments needed for function read.myacc.csv if you are working with a non-standard csv formatted files.

Value

P=P,filequality=filequality, switchoffLD = switchoffLD

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

Author(s)

Vincent T van Hees <vincentvanhees@gmail.com>

Examples

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

```
selectdaysfile=c(), filequality=filequality,
                        decn=".", dayborder=0,PreviousEndPage=c())

## End(Not run)
```

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

Description

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

Usage

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

Arguments

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

Value

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

Author(s)

Vincent T van Hees <vincentvanhees@gmail.com>

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

Description

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

Usage

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

Arguments

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

Value

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

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 <vincentvanhees@gmail.com>

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

Description

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

Usage

```
g.report.part5(metadatadir=c(),f0=c(),f1=c(),loglocation=c(),
                           includenightcrit=c(),includedaycrit=c(),
                           data_cleaning_file=c(),
                           includedaycrit.part5=2/3,
                           minimum_MM_length.part5=23)
```

Arguments

`metadatadir`	see g.part5
`f0`	see g.part5
`f1`	see g.part5
`loglocation`	see g.part4
`includenightcrit`	Despricated as of version 2.0, not used anymore in part 5 report
`includedaycrit`	Despricated 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.

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 <vincentvanhees@gmail.com>

`g.shell.GGIR` *Shell function for analysing an accelerometer dataset.*

Description

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

Usage

```
g.shell.GGIR(mode=1:5,datadir=c(),outputdir=c(),studyname=c(),
f0=1,f1=0, do.report=c(2),overwrite=FALSE,visualreport=FALSE,
viewingwindow=1,configfile=c(),myfun=c(),...)
```

Arguments

`mode`	Specify which of the four parts need to be run, e.g. mode = 1 makes that g.part1 is run. Default setting, mode = c(1,2), makes that both part1 and part2 are ran. Note that if mode = c(1,3) then the code will also set do.anglez = TRUE in order to enable sleep detection. If you run part 1 and 3 seperatedly then you need to remember to set argument do.anglez to TRUE when running part1.
`datadir`	Directory where the accelerometer files are stored or list, e.g. "C:/mydata" of accelerometer filenames and directories, e.g. c("C:/mydata/myfile1.bin", "C:/mydata/myfile2.bin").
`outputdir`	Directory where the output needs to be stored. Note that this function will attempt to create folders in this directory and uses those folder to keep output
`studyname`	If the datadir is a folder then the study will be given the name of the data directory. If datadir is a list of filenames then the studyname as specified by this input argument will be used as name for the study
`f0`	File index to start with (default = 1). Index refers to the filenames sorted in increasing order
`f1`	File index to finish with (defaults to number of files available)

overwrite	Do you want to overwrite analysis for which milestone data exists? If overwrite=FALSE then milestone data from a previous analysis will be used if available and visual reports will not be created again.
do.report	For which parts to generate a summary spreadsheet: 2 and/or 4. Default is c(2). A report will be generated based on the available milestone data. When creating milestone data with multiple machines it is advisable to turn the report generation off when generating the milestone data, value = c(), and then to merge the milestone data and turn report generation back on while setting overwrite to FALSE.
visualreport	If TRUE then generate visual report based on combined output from part 2 and 4. This is in beta-version at the moment.
viewingwindow	Centre the day as displayed around noon (value = 1) or around midnight (value = 2)
configfile	Configuration file previously generated by g.shell.GGIR. See details.
myfun	External function object to be applied to raw data, see g.getmeta.
...	Any input arguments needed for functions g.part1, g.part2, g.part3 or g.part4. See respective function documentation for further clarification.

Details

Once you have used g.shell.GGIR and the output directory (outputdir) will be filled with milestone data and results. Function g.shell.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 g.shell.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.

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 <vincentvanhees@gmail.com>

References

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

- van Hees VT, Fang Z, Langford J, Assah F, Mohammad A, da Silva IC, Trenell MI, White T, Wareham NJ, Brage S. Auto-calibration of accelerometer data for free-living physical activity assessment using local gravity and temperature: an evaluation on four continents. J Appl Physiol (1985). 2014 Aug 7
- van Hees VT, Sabia S, et al. (2015) A novel, open access method to assess sleep duration using a wrist-worn accelerometer, PLoS ONE, November 2015

Examples

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

```
                qwindow=c(0,24),
                ilevels = c(seq(0,400,by=50),8000),
                mvpathreshold =c(100,120),
                #------------------------------
                # Part 3 parameters:
                #------------------------------
                timethreshold= c(5,10),
                anglethreshold=5,
                ignorenonwear = TRUE,
                #------------------------------
                # Part 4 parameters:
                #------------------------------
                excludefirstlast = FALSE,
                includenightcrit = 16,
                def.noc.sleep = 1,
                loglocation= "D:/sleeplog.csv",
                outliers.only = FALSE,
                criterror = 4,
                relyonsleeplog = FALSE,
                sleeplogidnum = TRUE,
                colid=1,
                coln1=2,
                do.visual = TRUE,
                nnights = 9,
                #------------------------------
                # Part 5 parameters:
                #------------------------------
                # Key functions: Merging physical activity with sleep analyses
                threshold.lig = c(30,40,50),
                threshold.mod = c(100,120),
                threshold.vig = c(400,500),
                excludefirstlast = FALSE,
                boutcriter = 0.8,
                boutcriter.in = 0.9,
                boutcriter.lig = 0.8,
                boutcriter.mvpa = 0.8,
                boutdur.in = c(10,20,30),
                boutdur.lig = c(1,5,10),
                boutdur.mvpa = c(1,5,10),
                timewindow = c("WW"),
                #-----------------------------------
                # Report generation
                #------------------------------
                do.report=c(2,4,5))

  ## End(Not run)
```

---

g.sib.det *sustiained inactivty bouts detection*

Description

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

Usage

```
g.sib.det(M, IMP, I, twd=c(-12,12), anglethreshold = 5,
            timethreshold = c(5,10), acc.metric = "ENMO",
            desiredtz="", constrain2range=TRUE, dayborder=0,
            myfun=c())
```

Arguments

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

Value

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

Author(s)

Vincent T van Hees <vincentvanhees@gmail.com>

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

Description

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

Usage

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

Arguments

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

Value

Function has no output other than the plot

Author(s)

Vincent T van Hees <vincentvanhees@gmail.com>

g.sib.sum *sustiained inactivty bouts detection*

Description

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

Usage

```
g.sib.sum(SLE,M,ignorenonwear=TRUE,desiredtz="")
```

Arguments

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

Value

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

Author(s)

Vincent T van Hees <vincentvanhees@gmail.com>

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

Description

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

Usage

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

Arguments

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

Details

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

Value

`rawxyz`	matrix with raw x, y, and, z acceleration values
`header`	file header
`timestamps`	local timestamps for `rawxyz`

Author(s)

Vincent T van Hees <vincentvanhees@gmail.com>

`g.weardec` *Detects whether accelerometer is worn*

Description

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

Usage

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

Arguments

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

Value

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

Author(s)

Vincent T van Hees <vincentvanhees@gmail.com>

Examples

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

getFirstTimestamp *Extract first timestamp from GENEActiv file*

Description

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

Usage

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

Arguments

f	GENEActiv filename
p1	First value of timestamps object

Value

POSIX object withstarttime

Author(s)

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

getfolderstructure *Extracts folderstructure based on data directory.*

Description

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

Usage

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

Arguments

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

Value

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

Examples

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

## End(Not run)
```

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

Description

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

Usage

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

Arguments

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

Value

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

Author(s)

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

Examples

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

getStartEndNumeric *Generate start and end page of a day*

Description

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

Usage

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

Arguments

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

Value

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

Author(s)

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

Examples

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

## End(Not run)
```

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	Optional, please see read.myacc.csv
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 <vincentvanhees@gmail.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, selectdaysfile, P, header, desiredtz, sf, i,
datafile, ws2, starttime, wday, weekdays, wdayname)
```

Arguments

temp.available	Boolean whether temperate is available.
monc	See g.inspectfile
dformat	See g.dotorcomma
data	Data part of g.readaccfile output
selectdaysfile	See g.dotorcomma

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.

Author(s)

Vincent T van Hees <vincentvanhees@gmail.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,
                boutdur.mvpa,boutcriter.mvpa,
                boutdur.lig,boutcriter.lig,
                boutdur.in,boutcriter.in,
                ws3,bout.metric)
```

Arguments

ts	Data.frame with time series genrated in .gpart5
TRLi	
TRMi	
TRVi	

boutdur.mvpa
boutcriter.mvpa

boutdur.lig
boutcriter.lig
boutdur.in
boutcriter.in
ws3,bout.metric

Value

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

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)
```

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")
```

---

numUnpack *Simple function using Rcpp*

Description

Simple function using Rcpp

Usage

```
numUnpack(pack)
```

Arguments

pack	vector of integer

Examples

```
  ## Not run:
    numUnpack()

## End(Not run)
```

POSIXtime2iso8601 *Convert POSIX to iso8601 timestamp*

Description

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

Usage

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

Arguments

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

Author(s)

Vincent T van Hees <vincentvanhees@gmail.com>

Examples

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

## End(Not run)
```

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=c(), 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)
```

Arguments

rmc.file	Filename of file to be read.
rmc.nrow	Number of rows to read, same as nrow argument in read.csv and in fread.
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 origin), "character", or "ActivPAL" (exotic timestamp format only used in the ActivPAL activity monitor).

`rmc.format.time`
Format of timestamp, 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.

`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 expriments took place. If experiments took place in a different timezone, then use this argument for the timezone in whcih 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 the 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

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 <vincentvanhees@gmail.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)
```

`resample` *Simple function using Rcpp*

Description

Simple function using Rcpp

Usage

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

Arguments

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

Examples

```
## Not run:
resample()

## End(Not run)
```

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

Description

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

Usage

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

Arguments

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

Value

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

Index

∗Topic **datasets**
data.calibrate, 9
data.getmeta, 10
data.inspectfile, 10

applyExtFunction, 5, *7*, *35*, *45*, *51*, *74*

chartime2iso8601, 6
check_myfun, 7
create_test_acc_csv, 8
create_test_sleeplog_csv, 9
createConfigFile, 7

data.calibrate, 9
data.getmeta, 10
data.inspectfile, 10
datadir2fnames, 11

g.abr.day.names, 12
g.analyse, 12, *15–20*, *27*, *30*, *32*, *37*, *47–49*
g.analyse.avday, *20*
g.analyse.avday *(*g.analyse.avy*)*, 15
g.analyse.avy, 15
g.analyse.perday, 17, *19, 20*
g.analyse.perfile, 19
g.applymetrics, 21
g.binread, *4*, 22
g.calibrate, *4*, *9*, *13*, *20*, 23, *43–45*, *65*, *81*, *90*
g.create.sp.mat, 25
g.createcoordinates, 25
g.cwaread, *4*, 26
g.detecmidnight, *16–18*, 27
g.dotorcomma, 28, *32*, *36*, *66*, *81*
g.downsample, 29
g.extractheadervars, *18, 19*, 29
g.getbout, *14*, 30, *49*, *57*
g.getidfromheaderobject, 31
g.getM5L5, *14*, *16*, *18*, 32, *49*
g.getmeta, *6, 7*, *10*, *12–14*, *16*, *18*, *20*, *23*, *26*, *28*, 33, *37*, *39*, *43, 44*, *48–50*, *53*, *56, 57*, *63*, *65, 66*, *71*, *74, 75*, *77*, *81, 82*, *90*
g.getstarttime, 36
g.impute, *12, 13*, *16, 17*, *19, 20*, 37, *47–49*, *55*, *63*, *74*
g.inspectfile, *10*, *13*, *19*, *30*, *32*, *36, 37*, 38, *63*, *66*, *74, 75*, *81*
g.intensitygradient, 39
g.IVIS, *14*, *16*, *20*, 40, *49*
g.loadlog, 41
g.metric, *21*, 42
g.part1, *5*, *11*, *23*, *35, 36*, *43*, 43, *47*, *49–51*, *56*, *65, 66*, *70*, *74*, *77*, *79, 80*
g.part2, *27*, *43*, 47, *53*, *65–67*, *70*
g.part3, 50, *70*, *74, 75*
g.part4, *41*, 52, *60*, *68–70*
g.part5, 54, *58–62*, *65*, *69*
g.part5.addfirstwake, 58
g.part5.addsib, 59
g.part5.definedays, 59
g.part5.fixmissingnight, 60
g.part5.onsetwaketiming, 61
g.part5.savetimeseries, 62
g.part5.wakesleepwindows, 62
g.plot, *25*, 63
g.plot5, *12*, 64
g.readaccfile, *28*, *36*, 65, *81, 82*, *90*
g.report.part2, 67
g.report.part4, 68
g.report.part5, 69
g.shell.GGIR, *5*, *7*, *43*, *47*, *64, 65*, 70
g.sib.det, 73, *75*
g.sib.plot, 75
g.sib.sum, 75
g.wavread, *26*, 76
g.weardec, *16*, *18*, 77
get_nw_clip_block_params, 80
get_starttime_weekday_meantemp_truncdata, 81

getFirstTimestamp, 78
getfolderstructure, 78
getStartEnd, 79
getStartEndNumeric, 80
GGIR *(*GGIR-package*)*, 4
GGIR-package, 4

identify_levels, 82
is.ISO8601, 83
is_this_a_dst_night, 85
isfilelist, *11*, 84
iso8601chartime2POSIX, 84

numUnpack, 85

POSIXtime2iso8601, 86

read.myacc.csv, *28*, *39*, *45*, *66*, *81*, 87
resample, 89

updateBlocksize, 90