Package 'GGIR'

July 24, 2026

Type Package

Title Raw Accelerometer Data Analysis

Version 3.3-8

Date 2026-07-24

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

Description A tool to process and analyse data collected with wearable raw acceleration sensors as described in Migueles and colleagues (JMPB 2019), and van Hees and colleagues (JApplPhysiol 2014; PLoSONE 2015). The package has been developed and tested for binary data from 'GENEActiv' <https://activinsights.com/>, binary (.gt3x) and .csv-export data from 'Actigraph' <https://ametris.com/> devices, and binary (.cwa) and .csv-export 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. Also the package allows for external function embedding.

URL https://github.com/wadpac/GGIR/, https://wadpac.github.io/GGIR/

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

License Apache License (== 2.0) | file LICENSE

Suggests testthat, covr, knitr, rmarkdown, actilifecounts, readxl

Imports data.table, foreach, doParallel, signal, zoo, unisensR, ineq, methods, psych, irr, lubridate, GGIRread, ActCR, read.gt3x, arrow

Depends stats, utils, R (>= 3.5)

VignetteBuilder knitr

NeedsCompilation no

Author Vincent T van Hees [aut, cre],
Jairo H Migueles [aut] (ORCID: <https://orcid.org/0000-0003-0366-6935>),
Samuel T Afolabi [ctb] (ORCID: <https://orcid.org/0009-0002-7325-6405>),
Severine Sabia [ctb],
Matthew R Patterson [ctb],
Zhou Fang [ctb],
Joe Heywood [ctb],

Joan Capdevila Pujol [ctb],
Lena Kushleyeva [ctb],
Mathilde Chen [ctb],
Manasa Yerramalla [ctb],
Patrick Bos [ctb] (ORCID: <https://orcid.org/0000-0002-6033-960X>),
Taren Sanders [ctb],
Chenxuan Zhao [ctb],
Ian Meneghel Danilevicz [ctb] (ORCID:
<https://orcid.org/0000-0003-4541-0524>),
Victor Barreto Mesquita [ctb],
Gaia Segantin [ctb],
Xiaoyu Zong [ctb] (ORCID: <https://orcid.org/0000-0001-5646-710X>),
Medical Research Council UK [cph, fnd],
Accelting [cph, fnd],
French National Research Agency [cph, fnd]

Repository CRAN

Date/Publication 2026-07-24 17:30:17 UTC

Contents

GGIR-package . . . 3
ABI . . . 5
apply_cosinor_IS_IV_Analyses . . . 6
create_test_acc_csv . . . 6
create_test_sleeplog_csv . . . 7
data.calibrate . . . 8
data.getmeta . . . 9
data.inspectfile . . . 9
data.metalong . . . 10
data.ts . . . 10
DFA . . . 11
g.calibrate . . . 12
g.getbout . . . 13
g.getmeta . . . 14
g.imputeTimegaps . . . 16
g.inspectfile . . . 17
g.loadlog . . . 18
g.part1 . . . 19
g.part2 . . . 21
g.part3 . . . 22
g.part4 . . . 24
g.part5 . . . 26
g.part6 . . . 27
g.plot5 . . . 28
g.report.part2 . . . 30
g.report.part4 . . . 31
g.report.part5 . . . 32

g.report.part5_dictionary 33
g.report.part6 34
g.shell.GGIR 35
GGIR 35
inspect_binFile_brand 61
is.ISO8601 62
iso8601chartime2POSIX 63
load_params 63
part6AlignIndividuals 64
part6PairwiseAggregation 64
POSIXtime2iso8601 65
read.myacc.csv 65
SSP 69
visualReport 70
write_dashboard_parquet 71

Index **74**

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

Description

Disclaimer: If you are a new GGIR user then please see the GGIR github-pages for a narrative overview of 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:	3.3-8
Date:	2026-07-24
License:	Apache License (== 2.0)
Discussion group:	https://groups.google.com/forum/#!forum/rpackageggir

Author(s)

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

References

- Migueles JH, Rowlands AV, et al. GGIR: A Research Community-Driven Open Source R Package for Generating Physical Activity and Sleep Outcomes From Multi-Day Raw Accelerometer Data. Journal for the Measurement of Physical Behaviour. 2(3) 2019. doi:10.1123/jmpb.2018-0063.
- van Hees VT, Gorzelniak L, Dean Leon EC, Eder M, Pias M, et al. (2013) Separating Movement and Gravity Components in an Acceleration Signal and Implications for the Assessment of Human Daily Physical Activity. PLoS ONE 8(4): e61691. doi:10.1371/journal.pone.0061691
- van Hees VT, Fang Z, Langford J, Assah F, Mohammad A, da Silva IC, Trenell MI, White T, Wareham NJ, Brage S. Auto-calibration of accelerometer data for free-living physical activity assessment using local gravity and temperature: an evaluation on four continents. J Appl Physiol (1985). 2014 Aug 7
- van Hees VT, Sabia S, et al. (2015) A novel, open access method to assess sleep duration using a wrist-worn accelerometer, PLoS ONE, November 2015

Examples

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

  #autocalibration:
  C = g.calibrate(datafile)

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

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

  #impute meta-data:
```

```
IMP = g.impute(M = data.getmeta, I = data.inspectfile)
#analyse and produce summary:
A = g.analyse(I = data.inspectfile, C = data.calibrate, M = data.getmeta, IMP, ID = "01wk0")
#plot data
g.plot(IMP, M = data.getmeta, I = data.inspectfile, durplot=4)
```

ABI *Activity balance index (ABI)*

Description

This function estimates the Activity balance index (ABI), which is a transformation of the self-similarity parameter (SSP), also known as scaling exponent or alpha.

Usage

```
ABI(x)
```

Arguments

x	the estimated self-similarity parameter (SSP)

Details

ABI = exp(-abs(SSP-1)/exp(-2))

Value

The estimated Activity balance index (ABI) is a real number between zero and one.

Author(s)

Ian Meneghel Danilevicz <ian.meneghel-danilevicz@inserm.fr>

References

C.-K. Peng, S.V. Buldyrev, S. Havlin, M. Simons, H.E. Stanley, A.L. Goldberger Phys. Rev. E, 49 (1994), p. 1685 Mesquita, Victor & Filho, Florencio & Rodrigues, Paulo. (2020). Detection of crossover points in detrended fluctuation analysis: An application to EEG signals of patients with epilepsy. Bioinformatics. 10.1093/bioinformatics/btaa955.

Examples

```
# Estimate Activity balance index of a very known time series
# available on R base: the sunspot.year.
## Not run:
  ssp = SSP(sunspot.year)
  abi = ABI(ssp)

## End(Not run)
```

`apply_cosinor_IS_IV_Analyses` *Apply Cosinor Analyses to time series*

Description

Wrapper function around cosinor_IS_IV_Analyses that first prepares the time series before applying the cosinorAnlayses

Usage

```
apply_cosinor_IS_IV_Analyses(ts, qcheck, midnightsi, epochsizes, threshold = NULL)
```

Arguments

`ts`	Data.frame with timestamps and acceleration metric.
`qcheck`	Vector of equal length as number of rows in ts with value 1 for invalid timestamps, 0 otherwise.
`midnightsi`	Indices for midnights in the time series
`epochsizes`	Epoch size for ts and qcheck respectively
`threshold`	See cosinor_IS_IV_Analyses

Author(s)

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

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

Description

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

Usage

```
create_test_acc_csv(sf=3,Nmin=2000,storagelocation=c(),
                    start_time = NULL, starts_at_midnight = FALSE)
```

Arguments

sf	Sample frequency in Hertz, the default here is low to minimize file size
Nmin	Number of minutes (minimum is 720)
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
start_time	Start time of the recording, in the hh:mm:ss format.
starts_at_midnight	Boolean indicating whether the recording should start at midnight. Ignored if start_time is specified.

Value

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

Examples

```
## Not run:
  create_test_acc_csv()

## End(Not run)
```

---

create_test_sleeplog_csv

Creates csv sleeplog file for testing purposes

Description

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

Usage

```
create_test_sleeplog_csv(Nnights = 7, storagelocation = c(),
                         advanced = FALSE, sep = ",", begin_date = "2016/06/25",
                            type = "sleeplog")
```

Arguments

Nnights	Number of nights (minimum is 1)
storagelocation	Location where the test file named testfile.csv will be stored If no value is provided then the function uses the current working directory
advanced	Boolean to indicate whether to create an advanced sleeplog that also includes logs of nap times and nonwear

sep	Character to indicate the column separator of the csv file.
begin_date	Character to indicate first date (in format "2016/06/25") to be used in the advanced sleeplog format. Ignored when generated basic sleeplog format.
type	Character to indicate which type of advanced sleeplog to create, either "sleeplog" with "wakeup" and "onset" columns, "bedlog" with "bedstart" and "bedend" columns, or "both" with both.

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)
```

data.metalong *Metalong object as part of part 1 milestone data*

Description

data.metalong is example of the metalong data.frame stored g.part1

Usage

```
data(data.metalong)
```

Format

The format is: chr "data.metalong"

Source

The data was collected on one individual for testing purposes

Examples

```
data(data.metalong)
```

data.ts *Time series data.frame stored by part 5*

Description

data.ts is example of the data.frame stored g.part5

Usage

```
data(data.ts)
```

Format

The format is: chr "data.ts"

Source

The data was collected on one individual for testing purposes and matches the data in object data.metalong

Examples

```
data(data.ts)
```

DFA *Detrended Fluctuation Analysis*

Description

Detrended Fluctuation Analysis (DFA)

Usage

```
DFA(data, scale = 2^(1/8), box_size = 4, m = 1)
```

Arguments

data	Univariate time series (must be a vector or data frame)
scale	Specifies the ratio between successive box sizes (by default scale = 2^(1/8))
box_size	Vector of box sizes (must be used in conjunction with scale = "F")
m	An integer of the polynomial order for the detrending (by default m=1)

Details

The DFA fluctuation can be computed in a geometric scale or for different choices of boxes sizes.

Value

A matrix with two columns:

box	The box sizes (time scales) used for the analysis.
DFA	The corresponding detrended fluctuation amplitudes (Root Mean Square).

Note

It is not possible estimating alpha for multiple time series at once.

Author(s)

Ian Meneghel Danilevicz <ian.meneghel-danilevicz@inserm.fr> Victor Barreto Mesquita <victormesquita40@hotmail.com> Jairo H Migueles <jairo@jhmigueles.com>

References

C.-K. Peng, S.V. Buldyrev, S. Havlin, M. Simons, H.E. Stanley, A.L. Goldberger Phys. Rev. E, 49 (1994), p. 1685 Mesquita, Victor & Filho, Florencio & Rodrigues, Paulo. (2020). Detection of crossover points in detrended fluctuation analysis: An application to EEG signals of patients with epilepsy. Bioinformatics. 10.1093/bioinformatics/btaa955.

Examples

```
# Estimate self-similarity of a very known time series available
# on R base: the sunspot.year.
# Then the spend time with each method is compared.
## Not run:
  dfa = DFA(sunspot.year)

## End(Not run)
```

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

Description

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

Usage

```
g.calibrate(datafile, params_rawdata = c(), params_general = c(),
          params_cleaning = c(), inspectfileobject = c(), verbose = TRUE, ...)
```

Arguments

`datafile`	Name of accelerometer file
`params_rawdata`	See g.part1
`params_general`	See g.part1
`params_cleaning`	See g.part1
`inspectfileobject`	Output from the function g.inspectfile.
`verbose`	Boolean (default = TRUE). to indicate whether console message should be printed. Note that warnings and error are always printed and can be suppressed with suppressWarning() or suppressMessages().
`...`	Any argument used in the previous version of g.calibrate, which will now be used to overrule the arguments specified with the parameter objects.

Value

`scale`	scaling correction values, e.g. c(1,1,1)
`offset`	offset correction values, e.g. c(0,0,0)
`tempoffset`	correction values related to temperature, e.g. c(0,0,0)

cal.error.start	absolute difference between Euclidean norm during all non-movement windows and 1 g before autocalibration
cal.error.end	absolute difference between Euclidean norm during all non-movement windows and 1 g after autocalibration
spheredata	average, standard deviation, Euclidean norm and temperature (if available) for all ten second non-movement windows as used for the autocalibration procedure
npoints	number of 10 second no-movement windows used to populate the sphere
nhoursused	number of hours of measurement data scanned to find the ten second time windows with no movement
meantempcal	mean temperature corresponding to the data as used for autocalibration. Only applies to data where temperate data is collected and available to GGIR, such as GENEActiv, Axivity, and in some instances ad-hoc .csv data.

Author(s)

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

References

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

Examples

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

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

## End(Not run)
```

---

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

Value

Vector with binary numbers indicator where bouts where detected

Author(s)

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

Examples

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

---

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

Description

Reads a accelerometer file in blocks, extracts various features and stores average feature value per short or long epoch. Acceleration and angle metrics are stored at short epoch length. The non-wear indication score, the clipping score, temperature (if available), light (if available), and Euclidean norm are stored at long epoch length. The function has been designed and thoroughly tested with accelerometer files from GENEA and GENEActiv bin files. Further, the function should be able to cope with ActiGraph gt3x and csv files, Axivity cwa and csv files, Movisens bin files, and ad-hoc csv files read through the read.myacc.csv function.

Usage

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

Arguments

datafile	name of accelerometer file
params_metrics	See details in GGIR.
params_rawdata	See details in GGIR.
params_general	See details in GGIR.
params_cleaning	See details in GGIR.
daylimit	number of days to limit (roughly), if set to FALSE no daylimit will be applied
offset	offset correction value per axis, usage: value = scale(value,center = -offset, scale = 1/scale)
scale	scaling correction value per axis, usage: value = scale(value,center = -offset, scale = 1/scale)
tempoffset	temperature offset correction value per axis, usage: value = scale(value,center = -offset, scale = 1/scale) + scale(temperature, center = rep(averagetemperate,3), scale = 1/tempoffset)
meantempcal	mean temperature corresponding to the data as used for autocalibration. If autocalibration is not done or if temperature was not available then leave blank (default)
myfun	External function object to be applied to raw data. See details applyExtFunction.
inspectfileobject	Output from the function g.inspectfile.
verbose	Boolean (default = TRUE). to indicate whether console message should be printed. Note that warnings and error are always printed and can be suppressed with suppressWarning() or suppressMessages().
...	Any argument used in the previous version of g.getmeta, which will now be used to overrule the arguments specified with the parameter objects.

Value

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

Author(s)

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

References

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

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

Examples

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

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

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

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

## End(Not run)
```

`g.imputeTimegaps` *Impute gaps in three axis raw accelerometer data*

Description

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

Usage

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

Arguments

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

Value

List including: - x, data.frame based on input x with timegaps imputed (as default) or with recordings with 0 values in the three axes removed (if impute = FALSE) - QClog, data.frame with information on the number of time gaps found and the total time imputed in minutes

Author(s)

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

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

Description

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

Usage

```
g.inspectfile(datafile, desiredtz = "", params_rawdata = c(),
                          configtz = c(), ...)
```

Arguments

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

Value

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

Author(s)

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

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

Arguments

`loglocation`	Location of the spreadsheet (csv) with sleep log information. See package vignette for explanation on expected format
`coln1`	Column number in the sleep log spreadsheet where the onset of the first night starts
`colid`	Column number in the sleep log spreadsheet in which the participant ID code is stored (default = 1)
`sleeplogsep`	Value used as sep argument for reading sleeplog csv file, usually "," or ";". This argument has been deprecated.
`meta.sleep.folder`	Path to part3 milestone data, only specify if sleeplog is in advanced format.
`desiredtz`	See GGIR
`sleepwindowType`	See GGIR

Value

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

Author(s)

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

Examples

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

## End(Not run)
```

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

Description

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

Usage

```
g.part1(datadir = c(), metadatadir = c(), f0 = 1, f1 = c(),
        myfun = c(), params_metrics = c(), params_rawdata = c(),
        params_cleaning = c(), params_general = c(), verbose = TRUE, ...)
```

Arguments

`datadir`	Directory where the accelerometer files are stored, e.g. "C:/mydata", or list of accelerometer filenames and directories, e.g. c("C:/mydata/myfile1.bin", "C:/mydata/myfile2.bin").
`metadatadir`	Directory where the output needs to be stored. Note that this function will attempt to create folders in this directory and uses those folder to keep output.
`f0`	File index to start with (default = 1). Index refers to the filenames sorted in alphabetical order
`f1`	File index to finish with (defaults to number of files available, i.e., f1 = 0)
`myfun`	External function object to be applied to raw data. See details applyExtFunction.
`params_metrics`	See details in GGIR.
`params_rawdata`	See details in GGIR.
`params_cleaning`	See details in GGIR.
`params_general`	See details in GGIR.
`verbose`	See details in GGIR.

... If you are working with a non-standard csv formatted files, g.part1 also takes any input arguments needed for function read.myacc.csv and argument rmc.noise from get_nw_clip_block_params. First test these argument with function read.myacc.csv directly. To ensure compatibility with R scripts written for older GGIR versions, the user can also provide parameters listed in the params_ objects as direct argument.

Details

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

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

Value

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

Author(s)

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

References

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

Examples

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

## End(Not run)
```

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

Description

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

Usage

```
g.part2(datadir = c(), metadatadir = c(), f0 = c(), f1 = c(),
      myfun = c(), params_cleaning = c(), params_247 = c(),
      params_phyact = c(), params_output = c(), params_general = c(),
      verbose = TRUE, ...)
```

Arguments

datadir	Directory where the accelerometer files are stored, e.g. "C:/mydata", or list of accelerometer filenames and directories, e.g. c("C:/mydata/myfile1.bin", "C:/mydata/myfile2.bin").
metadatadir	Directory that holds a folder 'meta' and inside this a folder 'basic' which contains the milestone data produced by g.part1. The folderstructure is normally created by g.part1 and GGIR will recognise what the value of metadatadir is.
f0	File index to start with (default = 1). Index refers to the filenames sorted in alphabetical order
f1	File index to finish with (defaults to number of files available, i.e., f1 = 0)
myfun	External function object to be applied to raw data. See details applyExtFunction.
params_cleaning	See details in GGIR.
params_247	See details in GGIR.
params_phyact	See details in GGIR.
params_output	See details in GGIR.
params_general	See details in GGIR.
verbose	See details in GGIR.
...	To ensure compatibility with R scripts written for older GGIR versions, the user can also provide parameters listed in the params_ objects as direct argument.

Details

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

argument to both g.part2 and GGIR. Directly specified argument overrule the configuration file and default values.

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

Value

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

Author(s)

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

References

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

Examples

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

## End(Not run)
```

---

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

Description

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

Usage

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

Arguments

metadatadir	Directory that holds a folder 'meta' and inside this a folder 'basic' which contains the milestone data produced by g.part1. The folderstructure is normally created by g.part1 and GGIR will recognise what the value of metadatadir is.
f0	File index to start with (default = 1). Index refers to the filenames sorted in alphabetical order
f1	File index to finish with (defaults to number of files available, i.e., f1 = 0)
myfun	External function object to be applied to raw data. See details applyExtFunction.
params_sleep	See details in GGIR.
params_metrics	See details in GGIR.
params_output	See details in GGIR.
params_general	See details in GGIR.
verbose	See details in GGIR.
...	To ensure compatibility with R scripts written for older GGIR versions, the user can also provide parameters listed in the params_ objects as direct argument.

Details

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

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

Value

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

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

Author(s)

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

References

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

Examples

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

## End(Not run)
```

---

g.part4 *Labels detected sustained inactivity periods by g.part3 as either part of the Sleep Period Time window or not*

Description

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

Usage

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

Arguments

datadir	Directory where the accelerometer files are stored, e.g. "C:/mydata", or list of accelerometer filenames and directories, e.g. c("C:/mydata/myfile1.bin", "C:/mydata/myfile2.bin").
metadatadir	Directory that holds a folder 'meta' and inside this a folder 'basic' which contains the milestone data produced by g.part1. The folderstructure is normally created by g.part1 and GGIR will recognise what the value of metadatadir is.
f0	File index to start with (default = 1). Index refers to the filenames sorted in alphabetical order
f1	File index to finish with (defaults to number of files available, i.e., f1 = 0)

params_sleep	List of parameters used for sleep analysis (GGIR part 3, 4, and 5): see documentation g.part3.
params_metrics	List of parameters used for metrics extraction (GGIR part 1): see documentation g.part1.
params_cleaning	See details in GGIR.
params_output	See details in GGIR.
params_general	See details in GGIR.
verbose	See details in GGIR.
...	To ensure compatibility with R scripts written for older GGIR versions, the user can also provide parameters listed in the params_ objects as direct argument.

Value

The function does not produce values but generates an RData file in the milestone subfolder ms4.out which incudes a dataframe named `nightsummary`. This dataframe is used in g.report.part4 to create two reports one per night and one per person. See package vignette paragraph "Output part 4" for description of all the variables.

Author(s)

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

References

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

Examples

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

## End(Not run)
```

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

Description

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

Usage

```
g.part5(datadir = c(), metadatadir = c(), f0 = c(), f1 = c(),
                  params_sleep = c(), params_metrics = c(),
                  params_247 = c(), params_phyact = c(),
                  params_cleaning = c(), params_output = c(),
                  params_general = c(), verbose = TRUE, ...)
```

Arguments

datadir	Directory where the accelerometer files are stored, e.g. "C:/mydata", or list of accelerometer filenames and directories, e.g. c("C:/mydata/myfile1.bin", "C:/mydata/myfile2.bin").
metadatadir	Directory that holds a folder 'meta' and inside this a folder 'basic' which contains the milestone data produced by g.part1. The folderstructure is normally created by g.part1 and GGIR will recognise what the value of metadatadir is.
f0	File index to start with (default = 1). Index refers to the filenames sorted in alphabetical order
f1	File index to finish with (defaults to number of files available, i.e., f1 = 0)
params_sleep	See details in GGIR.
params_metrics	See details in GGIR.
params_247	See details in GGIR.
params_phyact	See details in GGIR.
params_cleaning	See details in GGIR.
params_output	See details in GGIR.
params_general	See details in GGIR.
verbose	See details in GGIR.
...	To ensure compatibility with R scripts written for older GGIR versions, the user can also provide parameters listed in the params_ objects as direct argument.

Value

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

Author(s)

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

Examples

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

## End(Not run)
```

`g.part6` *Perform temporal pattern analyses*

Description

This function aims to facilitate time-pattern analysis building on the labelled time series derived in GGIR part 5

Usage

```
g.part6(datadir = c(), metadatadir = c(), f0 = c(), f1 = c(),
                    params_general = c(), params_phyact = c(), params_247 = c(),
                    params_cleaning = c(), verbose = TRUE, ...)
```

Arguments

`datadir`	Directory where the accelerometer files are stored, e.g. "C:/mydata", or list of accelerometer filenames and directories, e.g. c("C:/mydata/myfile1.bin", "C:/mydata/myfile2.bin").
`metadatadir`	Directory that holds a folder 'meta' and inside this a folder 'basic' which contains the milestone data produced by g.part1. The folderstructure is normally created by g.part1 and GGIR will recognise what the value of metadatadir is.
`f0`	File index to start with (default = 1). Index refers to the filenames sorted in alphabetical order
`f1`	File index to finish with (defaults to number of files available, i.e., f1 = 0)
`params_general`	See details in GGIR.
`params_phyact`	See details in GGIR.
`params_247`	See details in GGIR.
`params_cleaning`	See details in GGIR.
`verbose`	See details in GGIR.
`...`	To ensure compatibility with R scripts written for older GGIR versions, the user can also provide parameters listed in the params_ objects as direct argument.

Value

The function does not produce values but generates an RData file in the milestone subfolder ms6.out which incudes ... (TO BE COMPLETED). This dataframe is used in g.report.part6 to create reports. See package vignette paragraph (TO BE COMPLETED) for description of all the variables.

Author(s)

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

Examples

```
## Not run:
  metadatadir = "C:/myfolder/meta"
  g.part6(metadatadir=metadatadir)

## End(Not run)
```

`g.plot5` *Generate user-friendly visual report. The first part of the report summarizes important daily metrics in bar plot format. The second part of the report shows the raw data and annotations in 24-hr periods. Angle-z is shown with sleep annotations during the SPT (sleep period time) window. ENMO is shown with daytime inactivity and PA (physical activity) annotations in the lower section of each 24-hr plot. The PA annotations are based on a 10 minute bout metric and 80 of a 10 minute bout of MVPA. Vigorous PA is a short window of time above threshold.vig that is part of a bout of MVPA. Light PA is a short window of time above threshold.lig that is part of a bout of light PA.*

Description

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

Usage

```
g.plot5(metadatadir = c(), dofirstpage = FALSE, viewingwindow = 1,
f0 = c(), f1 = c(), overwrite = FALSE, metric="ENMO",desiredtz = "",
threshold.lig = 30, threshold.mod = 100, threshold.vig = 400,
visualreport_without_invalid = TRUE, includedaycrit = 0.66, includenightcrit = 0.66,
verbose = TRUE)
```

Arguments

`metadatadir`	Directory that holds a folder 'meta' and inside this a folder 'basic' which contains the milestone data produced by g.part1. The folderstructure is normally created by g.part1 and GGIR will recognise what the value of metadatadir is.

dofirstpage	Boolean to indicate whether a first page with historgrams summarizing the whole measurement should be added
viewingwindow	See GGIR
f0	File index to start with (default = 1). Index refers to the filenames sorted in alphabetical order
f1	File index to finish with (defaults to number of files available, i.e., f1 = 0)
overwrite	See GGIR
metric	Which one of the metrics do you want to consider to describe behaviour. The metric of interest need to be calculated in M (see g.part1)
desiredtz	See GGIR
threshold.lig	See GGIR
threshold.mod	See GGIR
threshold.vig	See GGIR
visualreport_without_invalid	See GGIR
includenightcrit	See GGIR
includedaycrit	See GGIR
verbose	See GGIR

Value

No values, this function only generates a plot

Author(s)

Vincent T van Hees <v.vanhees@accelting.com> Matthew R Patterson <mpatterson@shimmersensing.com>

Examples

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

## End(Not run)
```

g.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(),
               store.long = FALSE, params_output, myfun = c(),
               verbose = TRUE, desiredtz = "")
```

Arguments

metadatadir	Directory that holds a folder 'meta' and inside this a folder 'basic' which contains the milestone data produced by g.part1. The folderstructure is normally created by g.part1 and GGIR will recognise what the value of metadatadir is.
f0	File index to start with (default = 1). Index refers to the filenames sorted in alphabetical order
f1	File index to finish with (defaults to number of files available, i.e., f1 = 0)
store.long	Booelean to indicate whether output should stored in long format in addition to default wide format. Automatically turned to TRUE if using day segmentation with qwindow.
params_output	Parameters object, see GGIR
myfun	(Optional) List as documented in GGIR. g.report.part2 uses this object to extract the name of the external function being used, such that this can be reused in the output filenames.
verbose	See details in GGIR.
desiredtz	See details in GGIR.

Value

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

Author(s)

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

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

Description

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

Usage

```
g.report.part4(datadir = c(), metadatadir = c(), f0 = c(),
  f1 = c(), data_cleaning_file = c(),
  params_sleep, params_output, verbose = TRUE)
```

Arguments

`datadir`	Directory where the accelerometer files are stored, e.g. "C:/mydata", or list of accelerometer filenames and directories, e.g. c("C:/mydata/myfile1.bin", "C:/mydata/myfile2.bin").
`metadatadir`	Directory that holds a folder 'meta' and inside this a folder 'basic' which contains the milestone data produced by g.part1. The folderstructure is normally created by g.part1 and GGIR will recognise what the value of metadatadir is.
`f0`	File index to start with (default = 1). Index refers to the filenames sorted in alphabetical order
`f1`	File index to finish with (defaults to number of files available, i.e., f1 = 0)
`data_cleaning_file`	see GGIR
`params_sleep`	Parameters object, see GGIR
`params_output`	Parameters object, see GGIR
`verbose`	See details in GGIR.

Value

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

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

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

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

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

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

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

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

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

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

Author(s)

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

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

Description

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

Usage

```
g.report.part5(metadatadir = c(), f0 = c(), f1 = c(), loglocation = c(),
                          params_cleaning = NULL,
                          LUX_day_segments = c(), params_output,
                          verbose = TRUE)
```

Arguments

`metadatadir`	Directory that holds a folder 'meta' and inside this a folder 'basic' which contains the milestone data produced by g.part1. The folderstructure is normally created by g.part1 and GGIR will recognise what the value of metadatadir is.
`f0`	File index to start with (default = 1). Index refers to the filenames sorted in alphabetical order
`f1`	File index to finish with (defaults to number of files available, i.e., f1 = 0)
`loglocation`	see g.part4
`params_cleaning`	See details in GGIR.
`LUX_day_segments`	see g.part5
`params_output`	Parameters object, see GGIR
`verbose`	See details in GGIR.

Value

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

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

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

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

Author(s)

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

`g.report.part5_dictionary`

Generate data dictionary for reports from milestone data produced by g.part5

Description

Creates a data dictionary with the definitions of the outcomes exported in the reports from milestone data produced by g.part5. Not intended for direct use by package user.

Usage

```
g.report.part5_dictionary(metadatadir, params_output)
```

Arguments

`metadatadir`	Directory that holds a folder 'meta' and inside this a folder 'basic' which contains the milestone data produced by g.part1. The folderstructure is normally created by g.part1 and GGIR will recognise what the value of metadatadir is.
`params_output`	Parameters object, see GGIR

Value

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

The following files are stored in the root of the results folder: part5_dictionary_daysummary_* part5_dictionary_personsummary_*

Author(s)

Vincent T van Hees <v.vanhees@accelting.com> Jairo Hidalgo Migueles <jairo@jhmigueles.com>

`g.report.part6` *Generate report from milestone data produced by g.part6*

Description

Creates report from milestone data produced by g.part6. Not intended for direct use by package user

Usage

```
g.report.part6(metadatadir = c(), f0 = c(), f1 = c(),
                          params_cleaning = NULL, params_output,
                          verbose = TRUE)
```

Arguments

`metadatadir`	Directory that holds a folder 'meta' and inside this a folder 'basic' which contains the milestone data produced by g.part1. The folderstructure is normally created by g.part1 and GGIR will recognise what the value of metadatadir is.
`f0`	File index to start with (default = 1). Index refers to the filenames sorted in alphabetical order
`f1`	File index to finish with (defaults to number of files available, i.e., f1 = 0)
`params_cleaning`	See details in GGIR.
`params_output`	Parameters object, see GGIR
`verbose`	See details in GGIR.

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: part6_summary.csv

See package vignette "HouseHoldCoanalysis".

Author(s)

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

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

Description

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

Usage

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

Arguments

...	Any of the parameters used by GGIR.

Value

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

Author(s)

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

GGIR *Shell function for analysing an accelerometer dataset.*

Description

This function is designed to help users operate all steps of the analysis. It helps to generate and structure milestone data, and produces user-friendly reports. The function acts as a shell with calls to g.part1, g.part2, g.part3, g.part4 and g.part5.

Usage

```
GGIR(mode = 1:5,
     datadir = c(),
     outputdir = c(),
     studyname = c(),
     f0 = 1, f1 = 0,
     do.report = c(2, 4, 5, 6),
     configfile = c(),
     myfun = c(),
     verbose = TRUE, ...)
```

Arguments

mode	Numeric (default = 1:5). Specify which of the five parts need to be run, e.g., mode = 1 makes that g.part1 is run; or mode = 1:5 makes that the whole GGIR pipeline is run, from g.part1 to g.part5. Optionally mode can also include the number 6 to tell GGIR to run g.part6 which is currently under development.
datadir	Character (default = c()). Directory where the accelerometer files are stored, e.g., \"C:/mydata\", or list of accelerometer filenames and directories, e.g. c(\"C:/mydata/myfile1.bin\", \"C:/mydata/myfile2.bin\").
outputdir	Character (default = c()). Directory where the output needs to be stored. Note that this function will attempt to create folders in this directory and uses those folder to keep output.
studyname	Character (default = c()). If the datadir is a folder, then the study will be given the name of the data directory. If datadir is a list of filenames then the studyname as specified by this input argument will be used as name for the study.
f0	Numeric (default = 1). File index to start with (default = 1). Index refers to the filenames sorted in alphabetical order.
f1	Numeric (default = 0). File index to finish with (defaults to number of files available).
do.report	Numeric (default = c(2, 4, 5, 6)). For which parts to generate a summary spreadsheet: 2, 4, 5, and/or 6. Default is c(2, 4, 5, 6). A report will be generated based on the available milestone data. When creating milestone data with multiple machines it is advisable to turn the report generation off when generating the milestone data, value = c(), and then to merge the milestone data and turn report generation back on while setting overwrite to FALSE.
configfile	Character (default = c()). Configuration file previously generated by function GGIR. See also https://wadpac.github.io/GGIR/articles/chapter2_Pipeline.html#configuration-file-
myfun	List (default = c()). External function object to be applied to raw data. See package vignette for detailed tutorial with examples on how to use the function embedding: https://wadpac.github.io/GGIR/articles/ExternalFunction.html
verbose	Boolean (default = TRUE). to indicate whether console message should be printed. Note that warnings and error are always printed and can be suppressed with suppressWarning() or suppressMessages().
...	Any of the parameters used GGIR. Given the large number of parameters used in GGIR we have grouped them in objects that start with \"params_\". These are documented in the details section. You cannot provide these objects as argument to function GGIR, but you can provide the parameters inside them as input to function GGIR.

Details

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

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

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

The parameters that can be used in GGIR are:

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

overwrite Boolean (default = FALSE). Do you want to overwrite analysis for which milestone data exists? If overwrite = FALSE, then milestone data from a previous analysis will be used if available and visual reports will not be created again.

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

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

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

acc.metric Character (default = \"ENMO\"). Which one of the acceleration metrics do you want to use for all acceleration magnitude analyses in GGIR part 5 and the visual report? For example: \"ENMO\", \"LFENMO\", \"MAD\", \"NeishabouriCount_y\", or \"NeishabouriCount_vm\". Only one acceleration metric can be specified and the selected metric needs to have been calculated in part 1 (see g.part1) via arguments such as `do.enmo = TRUE` or `do.mad = TRUE`.

part5_agg2_60seconds Boolean (default = FALSE). Whether to use aggregate epochs to 60 seconds as part of the GGIR g.part5 analysis. Aggregation is doen by averaging. Note that when working with count metrics such as Neishabouri counts this means that the threshold can stay the same as in part 2, because again the threshold is expressed relative to the original epoch size, even if averaged per minute. For example if we want to use a cut-point 100 count per minute then we specify `mvpathreshold = 100 * (5/60)` as well as 'threshold.mod = `100 * (5/60)` regardless of whether we set part5_agg2_60seconds to TRUE or FALSE.

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

desiredtz Character (default = \"\", i.e., system timezone). Timezone in which device was configured and experiments took place. If experiments took place in a different timezone, then use this argument for the timezone in which the experiments took place and argument `configtz` to specify where the device was configured. Use the \"TZ identifier\" as specified at https://en.wikipedia.org/wiki/Zone.tab to set desiredtz, e.g., \"Europe/London\". As of GGIR 3.2-12, `desiredtz` is only configurable in GGIR part 1 and passed on to the following parts

via de milestone data files stored by each part. This means that if you would run `GGIR(mode = 2, desiredtz = "Europe/Berlin")` GGIR would ignore the timezone specification and rely on the timezone specified when part 1 was processed specific to each recording. As a result, it is now possible to process part 1 separately with different timezones, then pool the milestone data into one output folder, and finally process all files jointly with the subsequent GGIR parts (2-6) which will automatically accounts for each individual's file timezone.

configtz Character (default = \"\", i.e., system timezone). At the moment only functional for GENEActiv .bin, AX3 cwa, ActiGraph .gt3x, and ad-hoc csv file format. 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. Use the \"TZ identifier\" as specified at https://en.wikipedia.org/wiki/Zone.tab to set configtz, e.g., \"Europe/London\".

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

windowsizes Numeric vector, three values (default = c(5, 900, 3600)). To indicate the lengths of the windows as in c(window1, window2, window3): window1 is the short epoch length in seconds, by default 5, and this is the time window over which acceleration and angle metrics are calculated; window2 is the long epoch length in seconds for which non-wear and signal clipping are defined, default 900 (expected to be a multitude of 60 seconds); window3 is the window length of data used for non-wear detection and by default 3600 seconds. So, when window3 is larger than window2 we use overlapping windows, while if window2 equals window3 non-wear periods are assessed by non-overlapping windows.

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

expand_tail_max_hours Numeric (default = NULL). This parameter has been replaced by `recordingEndSleepHour`.

recordingEndSleepHour Numeric (default = NULL). Time (in hours) at which the recording should end (or later) to expand the g.part1 output with synthetic data to trigger sleep detection for last night. Using argument `recordingEndSleepHour` implies the assumption that the participant fell asleep at or before the end of the recording if the recording ended at or after `recordingEndSleepHour` hour of the last day. This assumption may not always hold true and should be used with caution. The synthetic data for metashort entails: timestamps continuing regularly, zeros for acceleration metrics other than EN, one for EN. Angle columns are created in a way that it triggers the sleep detection using the equation: `round(sin((1:length_expansion) / (900/epochsize))) * 15`. To keep track of the tail expansion g.part1 stores the length of the expansion in the RData files, which is then passed via g.part2, g.part3, and g.part4 to g.part5. In g.part5 the tail expansion size is included as an additional variable in the csv-reports. In the g.part4 csv-report the last night is omitted, because we know that sleep estimates from the last night will not be trustworthy. Similarly, in the g.part5 output columns related to the sleep assessment will be omitted for the last window to avoid biasing the averages. Further, the synthetic data are also ignored in the visualizations and time series output to avoid biased output.

dataFormat Character (default = \"raw\"). To indicate what the format is of the data in datadir. Alternatives: ukbiobank_csv, actiwatch_csv, actiwatch_awd, actigraph_csv, sensewear_xls,

phb_xlsx, and fitbit_json which correspond to epoch level data files from, respecitively, UK Biobank in csv format, Actiwatch in csv format, Actiwatch in awd format, ActiGraph csv format, Sensewear in xls format (also works with xlsx), Philips Health Band in xlsx format, and Fitbit in json format. Here, the assumed epoch size for UK Biobank csvdata is 5 seconds. The epoch size for the other non-raw data formats is flexible, but make sure that you set first value of argument `windowsizes` accordingly. Also when working with non-raw data formats specify argument `extEpochData_timeformat` as documented below. For ukbiobank_csv nonwear is a column in the data itself, for actiwatch_csv, actiwatch_awd, actigraph_csv, and sensewear_xls non-wear is detected as 60 minute rolling zeros. The length of this window can be modified with the third value of argument `windowsizes` expressed in seconds.

maxRecordingInterval Numeric (default = NULL). To indicate the maximum gap in hours between repeated measurements with the same ID for the recordings to be appended. So, the assumption is that the ID can be matched, make sure argument `idloc` is set correctly. If argument `maxRecordingInterval` is set to NULL (default) recordings are not appended. If recordings overlap then GGIR will use the data from the latest recording. If recordings are separated then the timegap between the recordings is filled with data points that resemble monitor not worn. The maximum value of maxFile gap is 504 (21 days). Only recordings from the same accelerometer brand are appended. This functionality is applied after all other aspect of GGIR part 1 are completed for all input files. The part 2 csv report will show number of appended recordings, sampling rate for each, time overlap or gap and the names of the filenames of the respective recording.

recording_split_times Character (default = NULL). To indicate path to a csv file with a column that has ID in the column name that holds the participant IDs, followed by columns with either dates or full timestamps on which the recording should be split. Here, timestamp format is specified with parameter `recording_split_timeformat`. The names of these columns are used to refer to each resulting segment. To prevent extremely long file name GGIR ignores spaces and does not consider more than 10 characters which are forced to lower case. If the recording starts before or ends after the first and last split time then the start and/or end of the recording is also used as split time and referred to as startrec and endrec, respectively. Files are split after all other aspect of GGIR part 1 are completed for all input files.

recording_split_timeformat Character (default = \"%d/%m/%Y %H:%M\") To indicate timestamp format as used in file as specified with `recording_split_times`. For guidance on how to specify time formats in R see: https://wadpac.github.io/GGIR/articles/DateTimeFormatsInR.html.

recording_split_ignore_edges Boolean (default = FALSE) To indicate whether the recording time before and after the time range defined by `recording_split_times` should be considered as segments.

recording_split_overlap Numeric (default = 0). Number of hours to use as buffer when splitting recording. A possitive number means that the split recordings overlap, a nevative number indicates a gap.

extEpochData_timeformat Character (default = \"%d-%m-%Y %H:%M:%S\"). To specify the time format used in the external epoch level data when argument `dataFormat` is set to \"actiwatch_csv\", \"actiwatch_awd\", \"actigraph_csv\" or \"sensewear_xls\". For example \"%Y-%m-%d %I:%M:%S %p\" for \"2023-07-11 01:24:01 PM\" or \"%m/%d/%Y %H:%M:%S\" \"2023-07-11 13:24:01\". For guidance on how to specify time formats in R see: https://wadpac.github.io/GGIR/articles

use_trycatch_serial Boolean (default = FALSE). If set to TRUE and when using the non-default `do.parallel = FALSE`, the core processing function in each GGIR part is wrapped for each file in a `tryCatch` block. This allows GGIR to continue processing remaining files even if one or more files cause an error (e.g., due to file corruption). The error messages for failed files

are logged and printed in the console at the end of the processing. Note: these error messages are less detailed compared to when processing data with `use_trycatch_serial == FALSE`. So, to investigate an error message it may be advisable to set `use_trycatch_serial == FALSE`. Further, it should be noted that `tryCatch` blocks are always used when processing `do.parallel = TRUE`. The functionally described here is envisioned to be primarily useful when it not possible to process accelerometer files in parallel and there are many files. For example, when GGIR is expected to run only on a single thread in a computing cluster. In that scenario it would actually be better to investigate whether GGIR can be given more computing threads as GGIR is designed to coordinate the processing of files across processing threads. A second scenario where `use_trycatch_serial == TRUE` is useful is when processing files that are too large to be processed even if parameter `chunksize` is set to a minimal value. In that scenario you may want to set `use_trycatch_serial == TRUE` and `do.parallel = FALSE`. Note that under normal conditions we expect GGIR to not generate errors. If an error occurs, always investigate and report the error to the community to help make GGIR better. GGIR is free software and depends on you as community.

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

backup.cal.coef Character (default = \"retrieve\"). Option to use backed-up calibration coefficient instead of deriving the calibration coefficients when analysing the same file twice. Argument backup.cal.coef has two usecase. Use case 1: If the auto-calibration fails then the user has the option to provide back-up calibration coefficients via this argument. The value of the argument needs to be the name and directory of a csv-spreadsheet with the following column names and subsequent values: \"filename\" with the names of accelerometer files on which the calibration coefficients need to be applied in case auto-calibration fails; \"scale.x\", \"scale.y\", and \"scale.z\" with the scaling coefficients; \"offset.x\", \"offset.y\", and \"offset.z\" with the offset coefficients, and; \"temperature.offset.x\", \"temperature.offset.y\", and \"temperature.offset.z\" with the temperature offset coefficients. This can be useful for analysing short lasting laboratory experiments with insufficient sphere data to perform the auto-calibration, but for which calibration coefficients can be derived in an alternative way. It is the users responsibility to compile the csv-spreadsheet. Instead of building this file the user can also Use case 2: The user wants to avoid performing the auto-calibration repeatedly on the same file. If backup.cal.coef value is set to \"retrieve\" (default) then GGIR will look out for the \"data_quality_report.csv\" file in the outputfolder QC, which holds the previously generated calibration coefficients. If you do not want this happen, then deleted the data_quality_report.csv from the QC folder or set it to value \"redo\".

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

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

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

spherecrit Numeric (default = 0.3). The minimum required acceleration value (in g) on both sides of 0 g for each axis. Used to judge whether the sphere is sufficiently populated

minloadcrit Numeric (default = 168). The minimum number of hours the code needs to read for the autocalibration procedure to be effective (only sensitive to multitudes of 12 hrs, other values will be ceiled). After loading these hours only extra data is loaded if calibration error has not been reduced to under 0.01 g.

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

chunksize Numeric (default = 1). Value to specify the size of chunks to be loaded as a fraction of an approximately 12 hour period for auto-calibration procedure and as fraction of 24 hour period for the metric calculation, e.g., 0.5 equals 6 and 12 hour chunks, respectively. For machines with less than 4Gb of RAM memory or with < 2GB memory per process when using `do.parallel = TRUE` a value below 1 is recommended. The value is constrained by GGIR to not be lower than 0.05. Please note that setting 0.05 will not produce output when 3rd value of parameter windowsizes is 3600.

dynrange Numeric (default = NULL). Provide dynamic range of 8 gravity.

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

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

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

rmc.skip Numeric (default = 0). Number of rows to skip, same as skip argument in read.csv and in fread.

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

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

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

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

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

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

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

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

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

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

rmc.format.time Character (default = \"%Y-%m-%d %H:%M:%OS\"). Character giving a date-time format as used by strptime. Only used for rmc.unit.time: character and POSIX.. For guidance on how to specify time formats in R see: https://wadpac.github.io/GGIR/articles/DateTimeFormatsInR.html

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

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

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

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

rmc.desiredtz Character (default = NULL). Timezone in which experiments took place. This argument is scheduled to be deprecated and is now used to overwrite `desiredtz` if not provided.

rmc.configtz Character (default = NULL). Timezone in which device was configured. This argument is scheduled to be deprecated and is now used to overwrite `configtz` if not provided.

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

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

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

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

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

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

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

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

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

rmc.scalefactor.acc Numeric value (default 1) to scale the acceleration signals via multiplication. For example, if data is provided in m/s2 then by setting this to 1/9.81 we would derive gravitational units.

frequency_tol Number (default = 0.1) as passed on to readAxivity from the GGIRread package. Represents the frequency tolerance as fraction between 0 and 1. When the relative bias per

data block is larger than this fraction then the data block will be imputed by lack of movement with gravitational oriationed guessed from most recent valid data block. Only applicable to Axivity .cwa data.

nonwear_range_threshold Numeric (default 150) used to define maximum value range per axis for non-wear detection, used in combination with brand specific standard deviation per axis.

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

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

$$angleX = (\tan^{-1}\frac{acc_{rollmedian(x)}}{(acc_{rollmedian(y)})^2 + (acc_{rollmedian(z)})^2}) * 180/\pi$$

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

$$angleY = (\tan^{-1}\frac{acc_{rollmedian(y)}}{(acc_{rollmedian(x)})^2 + (acc_{rollmedian(z)})^2}) * 180/\pi$$

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

$$angleZ = (\tan^{-1}\frac{acc_{rollmedian(z)}}{(acc_{rollmedian(x)})^2 + (acc_{rollmedian(y)})^2}) * 180/\pi$$

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

do.brondcounts Boolean (default = FALSE). this option has been deprecated (October 2022) due to issues with the activityCounts package that we used as a dependency. If TRUE, calculated the metric via R package activityCounts. We called them BrondCounts because there are large number of activity counts in the physical activity and sleep research field. By calling them _brondcounts_ we clarify that these are the counts proposed by Jan Brønd and implemented in R by Ruben Brondeel. The _brondcounts_ are intended to be an imitation of the counts produced by one of the closed source ActiLife software by ActiGraph.

do.neishabouricounts Boolean (default = FALSE). If TRUE, calculates the metric via R package actilifecounts, which is an implementation of the algorithm used in the closed-source software ActiLife by ActiGraph (methods published in doi: 10.1038/s41598-022-16003-x). We use the name of the first author (instead of ActiLifeCounts) of the paper and call them NeishabouriCount under the uncertainty that ActiLife will implement this same algorithm over time. To use the Neishabouri counts for the physical activity intensity classification in part 5 (i.e., metric over the threshold.lig, threshold.mod, and threshold.vig would be applied), the `acc.metric` argument needs to be set as one of the following: \"NeishabouriCount_x\", \"NeishabouriCount_y\", \"NeishabouriCount_z\", \"NeishabouriCount_vm\" to use the counts in the x-, y-, z-axis or vector magnitude, respectively.

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

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

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

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

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

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

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

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

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

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

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

TimeSegments2ZeroFile Character (default = NULL). Takes path to a csv file that has columns \"windowstart\" and \"windowend\" to refer to the start and end time of a time windows in format \"2024-10-12 20:00:00\", and \"filename\" of the GGIR milestone data file without the \"meta_\" segment of the name. GGIR part 2 uses this to set all acceleration values to zero and the non-wear classification to zero (meaning sensor worn). Motivation: When the accelerometer is not worn during the night GGIR automatically labels them as invalid, while the user may like to treat them as zero movement. Disclaimer: This functionality was developed in 2019. With hindsight it is not generic enough and in need for revision. Please contact GGIR maintainers if you would like us to invest time in improving this functionality.

data_cleaning_file Character (default = NULL). Optional path to a csv file you create that holds four columns: ID, day_part5, relyonguider_part4, and night_part4. ID should hold the participant ID. Columns day_part5 and night_part4 allow you to specify which day(s) and night(s) need to be excluded from g.part5 and g.part4, respectively. When including multiple day(s)/night(s) create a new line for each day/night. So, this will be done regardless of whether the rest of GGIR thinks those day(s)/night(s) are valid. Column relyonguider_part4

allows you to specify for which nights g.part4 should fully rely on the guider. See also package vignette.

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

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

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

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

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

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

study_dates_file Character (default = c()). Full path to csv file containing the first and last date of the expected wear period for every study participant (dates are provided per individual). Expected format of the activity diary is: First column headers followed by one row per recording. There should be three columns: first column is recording ID, which needs to match with the ID GGIR extracts from the accelerometer file; second column should contain the first date of the study; and third column the last date of the study. Date columns should be by default in format \"23-04-2017\", or in the date format specified by argument `study_dates_dateformat` (below). If not specified (default), then GGIR would use the first and last day of the recording as beginning and end of the study. Note that these dates are used on top of the `data_masking_strategy` selected.

study_dates_dateformat Character (default = \"%d-%m-%Y\"). To specify the date format used in the `study_dates_file` as used by strptime. For guidance on how to specify time formats in R see: https://wadpac.github.io/GGIR/articles/DateTimeFormatsInR.html

strategy Deprecated and replaced by `data_masking_strategy`. If `strategy` is specified then its value is passed on and used for `data_masking_strategy`.

data_masking_strategy Numeric (default = 1). How to deal with knowledge about study protocol. data_masking_strategy = 1 means select data based on `hrs.del.start` and `hrs.del.end`. data_masking_strategy = 2 makes that only the data between the first midnight and the last midnight is used. data_masking_strategy = 3 selects the most active X days in the file where X is specified by argument `ndayswindow`, where the days are a series of 24-h blocks starting any time in the day (X hours at the beginning and end of this period can be deleted with arguments `hrs.del.start` and `hrs.del.end`) data_masking_strategy = 4 to only use the data after the first midnight. data_masking_strategy = 5 is similar to `data_masking_strategy` = 3, but it selects X complete calendar days where X is specified by argument `ndayswindow` (X hours at the beginning and end of this period can be deleted with arguments `hrs.del.start` and `hrs.del.end`).

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

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

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

ndayswindow Numeric (default = 7). If `data_masking_strategy` is set to 3 or 5, then this is the size of the window as a number of days. For data_masking_strategy 3 value can be fractional, e.g. 7.5, while for data_masking_strategy 5 it needs to be an integer.

includedaycrit.part5 Numeric (default = 2/3). Inclusion criteria used in part 5 for number of valid hours during the waking hours of a day, when value is smaller than or equal to 1 used as fraction of waking hours, when value above 1 used as absolute number of valid hours required. Do not confuse this argument with argument `includedaycrit` which is only used in GGIR part 2 and applies to the entire day.

includenightcrit.part5 Numeric (default = 0). Inclusion criteria used in part 5 for number of valid hours during the sleep period hours of a day (the night), when value is smaller than or equal to 1 used as fraction of sleep period hours, when value above 1 used as absolute number of valid hours required. Do not confuse this argument with argument `includenightcrit` which is only used in GGIR part 4 and applies to the entire 24 hour window from noon to noon or 6pm to 6pm.

segmentWEARcrit.part5 Numeric (default = 0.5). Fraction of `qwindow` segment expected to be valid in part 5, where 0.3 indicates that at least 30 percent of the time should be valid.

segmentDAYSPTcrit.part5 Numeric vector or length 2 (default = c(0.9, 0)). Inclusion criteria for the proportion of the segment that should be classified as day (awake) and spt (sleep period time) to be considered valid. If you are interested in comparing time spent in behaviour then it is better to set one of the two numbers to 0, and the other defines the proportion of the segment that should be classified as day or spt, respectively. The default setting would focus on waking hour segments and includes all segments that overlap for at least 90 percent with waking hours. In order to shift focus to the SPT you could use c(0, 0.9) which ensures that all segments that overlap for at least 90 percent with the SPT are included. Setting both to zero would be problematic when comparing time spent in behaviours between days or individuals: A complete segment would be averaged with an incomplete segments (someone going to bed or waking up in the middle of a segment) by which it is no longer clear whether the person is less active or sleeps more during that segment. Similarly it is not clear whether the person has more wakefulness during SPT for a segment or woke up or went to bed during the segment.

includedaycrit Numeric (default = 16). Minimum required number of valid hours in calendar day specific to analysis in part 2. If you specify two values as in c(16, 16) then the first value will be used in part 2 and the second value will be used in part 5 and applied as a criterion on the full part 5 window. Note that this is then applied in addition to parameter includedaycrit.part5 which only looks at valid data during waking hours.

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

nonWearEdgeCorrection Boolean (default = TRUE). If TRUE then the non-wear detection around the edges of the recording (first and last 3 hours) are corrected following description in vanHees2013 as has been the default since then. This functionality is advisable when working with sleep clinic or exercise lab data typically lasting less than a day.

nonwear_approach Character (default = \"2023\"). Whether to use the traditional version of the non-wear detection algorithm (nonwear_approach = \"2013\") or the new version (nonwear_approach = \"2023\"). The 2013 version would use the longsize window (windowsizes[3], one hour as default) to check the conditions for nonwear identification and would flag as nonwear the mediumsize window (windowsizes[2], 15 min as default) in the middle. The 2023 version differs in which it would flag as nonwear the full longsize window. For the 2013 method the longsize window is centered in the centre of the mediumsize window, while

in the 2023 method the longsizewindow is aligned with its left edge to the left edge of the mediumsize window.

includecrit.part6 Numeric (default = c(2/3, 2/3)) Vector of two with the minimum fraction of valid data required for day and spt time, respectively. This criteria is only used for circadian rhythm analysis.

nonwearFiltermaxHours Numeric (default = NULL). If not NULL, ignore detected nonwear periods that last shorter than nonwearFiltermaxHours during a window as defined by parameter `NonwearFilterWindow`. If NonwearFilterWindow is not provided (set to NULL as is the default) check whether `qwindow` is defined as activity diary file and whether it has timestamps that allow for defining a time in bed, SPT or lightsoff window. For this the code looks for column names in the diary with the terms inbed, sleeponset, or lightsout to define the start of the window. Further, to define the end of the window the code looks for columns outbed, wakeup, or lightsoff. If yes, it uses the largest of the windows that can be defined with these timestamps. If not, use midnight-6am as fall back setting. You may realise that these columns are the same as the advanced format for the sleelog accepted by GGIR part 4. So, if you already have an advanced format sleeplog for part 4 then you can provide this as value for the qwindow parameter. If you want to use `qwindow` to guide the nonwear filtering as described in here but do not want GGIR to use it for day segment analysis in part 2 or 5 then make sure the diary filename includes the word \"onlyfilter\" or \"filteronly\" to tell GGIR to only use qwindow for nonwear filtering. The main purpose of this functionality is to offer the option to ignore short lasting nonwear episodes during the night when there is suspicion that these are falsely detect, e.g. in individuals with extended motionless sleep periods caused by medication.

nonwearFilterWindow Numeric (default = NULL). Vector of length 2 to specify the start and end hour of the night to be used for the functionality as described above for NonwearFiltermaxHours.

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

mvpathreshold Numeric (default = NULL). Legacy parameter, if not provided GGIR uses the value of `threshold.mod` for this. Acceleration threshold for MVPA estimation in GGIR g.part2. This can be a single number or an vector of numbers, e.g., `mvpathreshold = c(100, 120)`. In the latter case the code will estimate MVPA separately for each threshold. If this variable is left blank, e.g., `mvpathreshold = c()`, then MVPA is not estimated.

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

boutcriter Numeric (default = NULL). Legacy parameter, if not provided GGIR uses the value of `boutcriter.mvpa` for this. A number between 0 and 1, it defines what fraction of a bout needs to be above the mvpathreshold, only used in GGIR g.part2.

threshold.lig Numeric (default = 40). In g.part5: Threshold for light physical activity to separate inactivity from light. Value can be one number or an vector of multiple numbers, e.g., `threshold.lig =c(30,40)`. If multiple numbers are entered then analysis will be repeated for each combination of threshold values. Threshold is applied to the first metric in the milestone data, so if you have only specified `do.enmo = TRUE` then it will be applied to ENMO.

threshold.mod Numeric (default = 100). In g.part5: Threshold for moderate physical activity to separate light from moderate. Value can be one number or an vector of multiple numbers, e.g., `threshold.mod = c(100, 120)`. If multiple numbers are entered then analysis will be

repeated for each combination of threshold values. Threshold is applied to the first metric in the milestone data, so if you have only specified `do.enmo = TRUE` then it will be applied to ENMO.

threshold.vig Numeric (default = 400). In g.part5: Threshold for vigorous physical activity to separate moderate from vigorous. Value can be one number or an vector of multiple numbers, e.g., `threshold.vig =c(400,500)`. If multiple numbers are entered then analysis will be repeated for each combination of threshold values. Threshold is applied to the first metric in the milestone data, so if you have only specified `do.enmo = TRUE` then it will be applied to ENMO.

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

boutdur.in Numeric (default = c(10, 20, 30)). Duration(s) of inactivity bouts in minutes to be extracted. Inactivity bouts are detected in the segments of the data which were not labelled as sleep or MVPA bouts. It will start with the identification of the longest to the shortest duration. In the default setting, it will start with the identification of 30 minute bouts, followed by 20 minute bouts in the rest of the data, and followed by 10 minute bouts in the rest of the data. Note that we use the term inactivity instead of sedentary behaviour for the lowest intensity level of behaviour. The reason for this is that GGIR does not attempt to classifying the activity type sitting at the moment, by which we feel that using the term sedentary behaviour would fail to communicate that.

boutdur.lig Numeric (default = c(1, 5, 10)). Duration(s) of light activity bouts in minutes to be extracted. Light activity bouts are detected in the segments of the data which were not labelled as sleep, MVPA, or inactivity bouts. It will start with the identification of the longest to the shortest duration. In the default setting, this will start with the identification of 10 minute bouts, followed by 5 minute bouts in the rest of the data, and followed by 1 minute bouts in the rest of the data.

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

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

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

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

part6_threshold_combi Character (default = NULL) to indicate the threshold combination derived in part 5 to be used for part 6. For example, \"40_100_120\". If left in default value GGIR will use the first threshold value from parameters `threshold.lig`, `threshold.mod`, and `threshold.vig`.

bout.metric Deprecated parameter that was previously used to choose which bout detection metric (algorithm) GGIR should use. GGIR now uses only one algorithm.

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

relyonguider Boolean (default = FALSE). Sustained inactivity bouts (sib) that overlap with the guider are labelled as sleep. If `relyonguider = FALSE` and the sib overlaps only partially with

the guider then it is the sib that defines the edge of the SPT window and not the guider. If relyonguider = TRUE and the sib overlaps only partially with the guider then it is the guider that defines the edge of the SPT window and not the sib. If participants were instructed NOT to wear the accelerometer during waking hours and ignorenonware=FALSE then set to relyonguider=TRUE, in all other scenarios set to FALSE.

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

def.noc.sleep Numeric (default = 1). The time window during which sustained inactivity will be assumed to represent sleep, e.g., def.noc.sleep = c(21, 9). This is only used if no sleep log entry is available. If left blank def.noc.sleep = c() then the 12 hour window centred at the least active 5 hours of the 24 hour period will be used instead. Here, L5 is hardcoded and will not change by changing argument winhr in function g.part2. If def.noc.sleep is filled with a single integer, e.g., def.noc.sleep=c(1) then the window will be detected with based on built in algorithms. See argument HASPT.algo from HASPT for specifying which of the algorithms to use.

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

nnights Numeric (default = NULL). This argument has been deprecated.

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

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

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

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

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

HASPT.algo Character (default = \"HDCZA\"). To indicate what algorithm should be used for the sleep period time detection. Default \"HDCZA\" is Heuristic algorithm looking at Distribution of Change in Z-Angle as described in van Hees et al. 2018. Other options included: \"HorAngle\", which is based on HDCZA but replaces non-movement detection of the HDCZA algorithm by looking for time segments where the angle of the longitudinal sensor axis has an angle relative to the horizontal plane between -45 and +45 degrees. And \"NotWorn\" which is also the same as HDCZA but looks for time segments when a rolling average of acceleration magnitude is below 5 per cent of its standard deviation, see Cookbook vignette in the Annexes of https://wadpac.github.io/GGIR/ for more detailed guidance on how to use \"NotWorn\".

HDCZA_threshold Numeric (default = c()) If HASPT.algo is set to \"HDCZA\" and HDCZA_threshold is NOT NULL, (e.g., HDCZA_threshold = 0.2), then that value will be used as threshold in the 6th step in the diagram of Figure 1 in van Hees et al. 2018 Scientific Report (doi: 10.1038/s41598-018-31266-z). However, doing so has not been supported by research yet and is only intended to facilitate methodological research, so we advise sticking with the default in line with the publication. Further, if HDCZA_threshold is set to a numeric vector of

length 2, e.g. c(10, 15), that will be used as percentile and multiplier for the above mentioned 6th step.

HASPT.ignore.invalid Boolean (default = FALSE). To indicate whether invalid time segments should be ignored in the heuristic guiders. If FALSE (default), the imputed angle or activity metric during the invalid time segments are used. If TRUE, invalid time segments are ignored (i.e., they cannot contribute to the guider). If NA, then invalid time segments are considered to be no movement segments and can contribute to the guider. Further, the guider name in the output will be shown with \"+invalid\" its end, e.g. \"HDCZA+invalid\", to reflect the NA setting. When HASPT.algo is \"NotWorn\", HASPT.ignore.invalid is automatically set to NA.

HASIB.algo Character (default = \"vanHees2015\"). To indicate which algorithm should be used to define the sustained inactivity bouts (i.e., likely sleep). Options: \"vanHees2015\", \"Sadeh1994\", \"Galland2012\", \"NotWorn\", \"Oakley1997\", \"data\". For details see vignette: https://wadpac.github.io/GGIR/articles/chapter8_SleepFundamentalsSibs.html

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

sleeplogsep Character (default = NULL). This argument is deprecated.

nap_model Character (default = NULL). To specify classification model. Currently the only option is \"hip3yr\", which corresponds to a model trained with hip data in 3-3.5 olds trained with parent diary data. This functionality is currently superseded by nap detection which is triggered by possible_nap_window and possible_nap_dur. Leave nap_model as NULL when using the new functionality. More documentation to follow in 2025 when development work has completed.

longitudinal_axis Integer (default = NULL). To indicate which axis is the longitudinal axis. If not provided, the function will estimate longitudinal axis as the axis with the highest 24 hour lagged autocorrelation. Only used when sensor.location = \"hip\" or HASPT.algo = \"HorAngle\".

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

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

possible_nap_window Numeric (default = NULL). Numeric vector of length two with range in clock hours during which naps are assumed to take place, e.g., possible_nap_window = c(9, 18). The nap classification is only applied if both possible_nap_window and possible_nap_dur are specified. More documentation to follow in 2025 when development work has completed.

possible_nap_dur Numeric (default = NULL). Numeric vector of length two with range in duration (minutes) of a nap, e.g., possible_nap_dur = c(15, 240). The nap classification is only applied if both possible_nap_window and possible_nap_dur are both specified. More documentation to follow in 2025 when development work has completed.

sleepefficiency.metric Numeric (default = 1). If 1 (default), sleep efficiency is calculated as detected sleep time during the SPT window divided by log-derived time in bed. If 2, sleep efficiency is calculated as detected sleep time during the SPT window divided by detected duration in sleep period time plus sleep latency (where sleep latency refers to the difference

between time in bed and sleep onset). sleepefficiency.metric is only considered when parameter `sleepwindowType = \"TimeInBed\"`

possible_nap_edge_acc Numeric (default = Inf). Maximum acceleration before or after the SIB for the nap to be considered. By default this will allow all possible naps.

possible_nap_gap Numeric (default = 0). Time gap expressed in seconds that is allowed between the sustained inactivity bouts that form the naps.

sib_must_fully_overlap_with_TimeInBed Boolean (default = c(TRUE, TRUE)). To indicate whether sib must fully overlap with TimeInBed to be considered sleep for the start and end of time in bed, respectively. Only considered when parameter `sleepwindowType = \"TimeInBed\"`. Note that negative sleep latency, if any, will be reported in the night summary report (part4_nightsummary_sleep csv files) but these negative sleep latency and corresponding sleep efficiency values when calculating the person summary aggregate as stored in the person level report (part4_summary_sleep csv files).

oakley_threshold Numeric (default = 20) Threshold as used by the Oakley algorithm. Original documentation suggestion to choose between 20, 40 or 80, while in GGIR this can be any absolute number.

consider_marker_button Boolean (default = FALSE) Whether to consider the marker button as guider. Currently only functional for Actiwatch-type count-accelerometers such as Philips Health Band and MotionWatch 8. For details see: https://wadpac.github.io/GGIR/articles/chapter9_SleepFundamentals

impute_marker_button Boolean (default = FALSE) Whether to impute marker buttons on other days of the recording in the context of using it as guider with `consider_marker_button`. For details see https://wadpac.github.io/GGIR/articles/chapter9_SleepFundamentalsGuiders.html

nap_markerbutton_method Numeric (default = 0) Integer to indicate whether and how to use marker button for nap detection: 0 = do not use marker button for nap detection (default); 1 = if marker button is available use it for nap detection, if not rely on accelerometer. 2 = nearby marker button is condition for nap detection but rely on accelerometer to define exact nap timing, and; 3 = nearby marker button is condition for nap detection and used instead of accelerometer classification.

nap_markerbutton_max_distance Numeric (default = 30) When using `nap_markerbutton_method` with a value other than 0 `nap_markerbutton_max_distance` sets the maximum distance in minutes between the nearest marker button and the edge of a sustained inactivity bout. If the marker button lies inside the sustained inactivity bout but beyond the midpoint, no maximum distance is applied. For example, if a sustained activity bout lasts from 11:00:00 to 13:00:00 then its midpoint will be 12:00:00 and with the default `nap_markerbutton_max_distance` any marker button between 10:30:00 and 12:00:00 will be considered valid for the start of the nap and any marker button between 12:00:00 and 13:30:00 will be considered valid for the end of the nap.

SRI1_smoothing_wsize_hrs Numeric (default = NULL) As used for Sleep Regularity Calculation in g.part3. If `SRI1_smoothing_wsize_hrs` and `SRI1_smoothing_frac` are both specified, this is the windowsize for smoothing of detected sustained inactivity bouts before calculating SRI1. For example, if set to 1 and 0.8, we apply a rolling 1 hour window where each window with 80% or more sleep is classificed as sleep.

SRI1_smoothing_frac Numeric (default = NULL) As used for Sleep Regularity Calculation in g.part3. If `SRI1_smoothing_wsize_hrs` and `SRI1_smoothing_frac` are both specified, this is the fraction used to smooth detected sustained inactivity bouts before calculating SRI1. For example, if set to 1 and 0.8, we apply a rolling 1 hour window where each window with 80% or more sleep is classificed as sleep.

spt_min_block_dur Numeric (default = 30). Minimum size in minutes of a resting block to be considered part of SPT in the estimation of the guider window (relevant for HASPT.algo settings: HDCZA, HorAngle and NotWorn).

spt_max_gap_dur Numeric (default = 60). Maximum size in minutes of a gap between resting blocks to be considered part of SPT in the estimation of the guider window (relevant for HASPT.algo settings: HDCZA, HorAngle and NotWorn).

spt_max_gap_ratio Numeric (default = 1). Only considered when set to a value less than 1 Maximum ratio between gap duration and either rest block next to it. If the ratio for both is less than spt_max_gap_ratio the gap is merged with the neighbouring rest blocks and together they are considered as estimation of the SPT guider window (relevant for HASPT.algo settings: HDCZA, HorAngle and NotWorn).

HorAngle_threshold Numeric (default = 60). Angle threshold used for the HorAngle algorithm.

guider_cor_maxgap_hrs Numeric (default = 2). Maximum gap duration in hours as used for the correction method.

guider_cor_min_frac_sib Numeric (default = 0.5) Required fraction of guider-based rest period with sustained inactivity bouts (sib) to be considered as possible extension of guider.

guider_cor_min_hrs Numeric (default = 2) Minimum guider-based rest duration (hours) to be considered as possible extension of guider.

guider_cor_meme_frac_out Numeric (default = 0.9) Minimum required fraction of initial guider classification outside the median start to median end (abbreviated as meme) of the guider across days to initiate search for secondary that falls within this window.

guider_cor_meme_frac_in Numeric (default = 0.4) Minimum required fraction of alternative guider classification inside the median start to median end (abbreviated as meme) of the guider across days to be considered the new guider window.

guider_cor_meme_min_hrs Numeric (default = 1) Minimum required duration of the alternative guider classification inside the median start to median end (abbreviated as meme) of the guider across days to be considered the new guider window.

guider_cor_do Boolean (default = FALSE) Whether to apply the guider correction algorithm as discussed in Chapter 9 of the online documentation: https://wadpac.github.io/GGIR/articles/chapter9_SleepFundament
In short, when sleep is fragmented, algorithms to guide the detection of SPT such as HDCZA may occassionally miss a part of the night or incorrectly consider a day time nap as the main sleep window in a day. This optional algorithm aims to correct for such misclassifications.

guider_cor_meme_min_dys Numeric (default = 3) Minimum number of days required to consider median-median based step in the guider correction algorithm as controlled by parameter `guider_cor_meme_frac_out`, `guider_cor_meme_frac_in`, and `guider_cor_meme_min_hrs`. This parameter is set to 3 by default as a median becomes only meaningful with at least three values.

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

qwindow Numeric or character (default = c(0, 24)). To specify windows over which all variables are calculated, e.g., acceleration distribution, number of valid hours, LXMX analysis, MVPA. If numeric, qwindow should have length two, e.g., `qwindow = c(0, 24)`, all variables will only be calculated over the full 24 hours in a day. If `qwindow = c(8, 24)` variables will be calculated over the window 0-8, 8-24 and 0-24. All days in the recording will be segmented based on these values. If you want to use a day specific segmentation in each day

then you can set qwindow to be the full path to activity diary file (character). Expected format of the activity diary is: First column headers followed by one row per recording, first column is recording ID, which needs to match with the ID GGIR extracts from the accelerometer file. Followed by date column in format \"23-04-2017\", where date format is specified by parameter `qwindow_dateformat` (below). Use the character combination date, Date or DATE in the column name. This is followed by one or multiple columns with start times for the activity types in that day format in hours:minutes:seconds. The header of the column will be used as label for each activity type. Insert a new date column before continuing with activity types for next day. Leave missing values empty. If an activity log is used then individuals who do not appear in the activity log will still be processed with value `qwindow = c(0, 24)`. Dates with no activity log data can be skipped, no need to have a column with the date followed by a column with the next date. If times in the activity diary are not multiple of the short window size (epoch length), the next epoch is considered (e.g., with epoch of 5 seconds, 8:00:02 will be redefined as 8:00:05 in the activity log). When using the qwindow functionality in combination with GGIR part 5 then make sure to check that parameters `segmentWEARcrit.part5` and `segmentDAYSPTcrit.part5` are specified to your research needs. When using an activity diary be aware that any column name including the words \"impute\" or \"uncertain\" will be ignored. This means you can, for you own convenience, add columns to log which timestamps have been manually imputed or are considered uncertain.

qwindow_dateformat Character (default = \"%d-%m-%Y\"). To specify the date format used in the activity log as used by strptime. For guidance on how to specify time formats in R see: https://wadpac.github.io/GGIR/articles/DateTimeFormatsInR.html

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

winhr Numeric (default = 5). Vector of window size(s) (unit: hours) of LX and MX analysis, where look for least and most active consecutive number of X hours.

qlevels Numeric (default = NULL). Vector 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. Parameter qlevels can for example be used for the MX-metrics (e.g. Rowlands et al) as discussed in https://wadpac.github.io/GGIR/articles/chapter7_DescribingDataWithoutl of-quantiles-mx-metrics-by-rowlands-et-al-

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

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

IVIS_windowsize_minutes This argument has been deprecated.

IVIS_epochsize_seconds Numeric (default = NULL). This parameter has been deprecated.

IVIS.activity.metric This argument has been deprecated.

IVIS_acc_threshold This argument has been deprecated.

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

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

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

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

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

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

L5M5window Has been deprecated after version 1.5-24. This parameter 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 parameter qwindow.

cosinor Boolean (default = FALSE). Whether to apply the circadian rhythm analysis in part 2. This parameter will be superseded by new parameter `part2CR` to better reflect that this controls all circadian rhythm parameters. If `part2CR` is specified then that setting overwrites `cosinor`.

part2CR Boolean (default = FALSE). Whether circadian rhythm analysis should be run in part 2, including: cosinor analysis, extended cosinor analysis, IS, IV, and phi.

part6CR Boolean (default = FALSE) to indicate whether circadian rhythm analysis should be run by part 6, this includes: cosinor analysis, extended cosinor analysis, IS, IV, and phi. Optionally this can be expanded with detrended fluctutation analysis which is controlled by parameter 'part6DFA'.

part6HCA Boolean (default = FALSE) to indicate whether Household Co Analysis should be run by part 6.

part6Window Character vector with length two (default = c(\"start\", \"end\")) to indicate the start and the end of the time series to be used for circadian rhythm analysis in part 6. In other words, this parameters is not used for Household co-analysis. Alternative values are: \"Wx\", \"Ox\", \"Hx\", where \"x\" is a number to indicat the xth wakeup, onset or hour of the recording. Negative values for \"x\" are also possible and will count relative to the end of the recording. For example, c(\"W1\", \"W-1\") goes from the first till the last wakeup, c(\"H5\", \"H-5\") ignores the first and last 5 hours, and c(\"O2\", \"W10\") goes from the second onset till the 10th wakeup time.

part6DFA Boolean (default = FALSE) to indicate whether to perform Detrended Fluctuation Analysis. Turned off by default because it can be time consuming.

clevels Not fully operational yet, to be actived in 2025. Numeric vector (default = c(30, 150)) with length 2 to indicate cadence ranges used when summarising step counts, if available. When set to NULL, step counts will not be summarised in part 5 output.

SRI2_WASOmin Numeric (default = 30) Minimum WASO duration in minutes as used for Sleep Regularity Calculation in g.part6.

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

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

do.part2.pdf Boolean (default = TRUE). Now replaced by do.part2.png. If specified and do.part2.png is not specified then value will be assigned to do.part2.png. Used in g.part2 to indicate whether to generate a pdf before, but this is now a png file.

do.part2.png Boolean (default = TRUE). Used in g.part2 to indicate whether to generate a png file for each recording.

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

timewindow Character (default = c(\"MM\", \"WW\")). In g.part5: Timewindow over which summary statistics are derived. Value can be \"MM\" (midnight to midnight), \"WW\" (waking time to waking time), \"OO\" (sleep onset to sleep onset), or any combination of them.

save_ms5rawlevels Boolean (default = TRUE). In g.part5: Whether to save the time series classification (levels) as csv or RData files (as defined by save_ms5raw_format). Note that time stamps will be stored in the column timenum in UTC format (i.e., seconds from 1970-01-01). To convert timenum to time stamp format, you need to specify your desired time zone, e.g., as.POSIXct(mdat$timenum, tz = \"Europe/London\"). If you are not using GGIR part 6, are not interested in the visualreport generation, and not interested in time series then you may want to consider setting this parameter to FALSE.

save_ms5raw_format Character (default = \"RData\"). In g.part5: To specify how data should be stored: \"csv\", \"RData\", or both via c(\"csv\", \"RData\"). Only used if save_ms5rawlevels = TRUE.

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

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

do.sibreport Boolean (default = TRUE). In g.part4: To indicate whether to generate report for the sustained inactivity bouts (SIB). If set to TRUE and when an advanced sleep diary is available in part 4 then part 5 will use this to generate summary statistics on the overlap between self-reported nonwear and napping with SIB. Here, SIB can be filter based on parameter possible_nap_edge_acc and the first value of possible_nap_dur

do.visual Boolean (default = TRUE). In g.part4: If TRUE, the function will generate a pdf with a visual representation of the overlap between the sleeplog entries and the accelerometer detections. This can be used to visually verify that the sleeplog entries do not come with obvious mistakes.

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

criterror Numeric (default = 3). In g.part4: Only used if do.visual = TRUE and outliers.only = TRUE. criterror specifies the number of minimum number of hours difference between sleep log and accelerometer estimate for the night to be included in the visualisation.

visualreport Boolean (default = TRUE). If TRUE, store a visualreport in outputfolder 'results/file summary reports' for each recording. Note that this report was introduced in release 3.1-8 and replaced an older visual report that had been in GGIR for many years. The older report was only intended as a quick attempt to have something visual to show to study participants back in 2015. It was not designed for data quality checking purposes. The new visual report is specifically aimed at supporting data quality checks as it shows the time series data as generated and used in GGIR part 5. .

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

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

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

sep_reports Character (default = \",\"). Value used as sep parameter in fwrite for writing csv reports.

dec_reports Character (default = \".\"). Value used as dec parameter in fwrite for writing csv reports.

sep_config Character (default = \",\"). Value used as sep parameter in fwrite for writing csv config file.

dec_config Character (default = \".\"). Value used as dec parameter in fwrite for writing csv config file.

visualreport_without_invalid Boolean (default = TRUE). If TRUE, then reports generated with `visualreport = TRUE` named \"old_report_\" only show the windows with sufficiently valid data according to `includedaycrit` when viewingwindow = 1 or `includenightcrit` when viewingwindow = 2

require_complete_lastnight_part5 Boolean (default = FALSE). When set to TRUE: The last WW window is excluded if the recording ends between midnight and 3pm, and starts on a date that is on or one day before the recording end date; The last OO and MM window are excluded if recording ends between midnight and 9am, and starts on a date that is on or one day before the recording end date. This to avoid risk that recording end biases the sleep estimates for the last night.

old_visualreport Boolean (default = FALSE). If TRUE, then generate old visual report in addition to new visualreport. The old visualreport will eventually be deprecated and is not suitable for data quality assessment, see documentation for `visualreport` above for details.

visualreport_hrsPerRow Numeric (default = 36). Width of the plots in the new visualreport (named \"report_...\") expressed in hours. Expected to be in the range 24-48. If more than 24 then the extra time overlaps with the beginning of the next plot.

visualreport_focus Character (default = \"day\") Whether new visual report (names start with \"report...\") focuss on day or night. If set ot \"night\" the focus will be on the night.

visualreport_validcrit Numeric (default = 0). Value between 0 and 1 representing the fraction of data in a plot that is expected to be valid. If this criteria is not met the row is skipped. Only used in new visual report (names start with \"report...\").

method_research_vars Character (default = NULL). Vector with names of methodological variable categories to store in the csv output files. Currently on \"nap\" is available which affects the part5 output. The variables are intended for methodological research only and are by default turned off.

save_dashboard_parquet Boolean (default = FALSE). If TRUE, GGIR will export a consolidated Parquet file named '`ggir_results.parquet`' to the '`results/`' subfolder of the output directory after all requested GGIR parts have completed. The file merges the Part 5 day summary, Part 4 night summary, Part 2 day and person summaries, and the data quality report

into a single row-per-day table, and embeds per-day epoch-level time series as a nested list-column (epochs). Key-value Parquet metadata is attached to record the variable dictionary, activity threshold configuration, and accelerometer metric used. This output is intended for use with DuckDB-WASM powered dashboards. Requires the **arrow** package to be installed.

Value

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

Author(s)

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

References

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

Examples

```
## Not run:
  mode = c(1,2,3,4,5)
  datadir = \"C:/myfolder/mydata\"
  outputdir = \"C:/myresults\"
  studyname =\"test\"
  f0 = 1
  f1 = 2
  GGIR(#-------------------------------
       # General parameters
       #-------------------------------
       mode = mode,
       datadir = datadir,
       outputdir = outputdir,
       studyname = studyname,
       f0 = f0,
       f1 = f1,
       overwrite = FALSE,
       do.imp = TRUE,
       idloc = 1,
       print.filename = FALSE,
       storefolderstructure = FALSE,
       #-------------------------------
```

```
# Part 1 parameters:
#------------------------------
windowsizes = c(5,900,3600),
do.cal = TRUE,
do.enmo = TRUE,
do.anglez = TRUE,
chunksize = 1,
printsummary = TRUE,
#------------------------------
# Part 2 parameters:
#------------------------------
data_masking_strategy = 1,
ndayswindow = 7,
hrs.del.start = 1,
hrs.del.end = 1,
maxdur = 9,
includedaycrit = 16,
L5M5window = c(0,24),
M5L5res = 10,
winhr = c(5,10),
qlevels = c(c(1380/1440),c(1410/1440)),
qwindow = c(0,24),
ilevels = c(seq(0,400,by=50),8000),
mvpathreshold = c(100,120),
#------------------------------
# Part 3 parameters:
#------------------------------
timethreshold = c(5,10),
anglethreshold = 5,
ignorenonwear = TRUE,
#------------------------------
# Part 4 parameters:
#------------------------------
excludefirstlast = FALSE,
includenightcrit = 16,
def.noc.sleep = 1,
loglocation = \"D:/sleeplog.csv\",
outliers.only = FALSE,
criterror = 4,
relyonguider = FALSE,
colid = 1,
coln1 = 2,
do.visual = TRUE,
#------------------------------
# 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))

# For externally derived Actiwatch data in .AWD format:
GGIR(datadir = \"/media/actiwatch_awd\", # folder with epoch level .AWD file
   outputdir = \"/media/myoutput\",
   dataFormat = \"actiwatch_awd\",
   extEpochData_timeformat = \"%m/%d/%Y %H:%M:%S\",
   mode = 1:5,
   do.report = c(2, 4, 5),
   windowsizes = c(60, 900, 3600), # 60 is the expected epoch length
   visualreport = FALSE,
   outliers.only = FALSE,
   overwrite = TRUE,
   HASIB.algo = \"Sadeh1994\",
   def.noc.sleep = c()) # <= because we cannot use HDCZA for ZCY

# For externally derived Actiwatch data in .CSV format:
GGIR(datadir = \"/media/actiwatch_csv\", # folder with epoch level .AWD file
   outputdir = \"/media/myoutput\",
   dataFormat = \"actiwatch_csv\",
   extEpochData_timeformat = \"%m/%d/%Y %H:%M:%S\",
   mode = 1:5,
   do.report = c(2, 4, 5),
   windowsizes = c(15, 900, 3600), # 15 is the expected epoch length
   visualreport = FALSE,
   outliers.only = FALSE,
   HASIB.algo = \"Sadeh1994\",
   def.noc.sleep = c()) # <= because we cannot use HDCZA for ZCY

# For externally derived UK Biobank data in .CSV format:
GGIR(datadir = \"/media/ukbiobank\",
    outputdir = \"/media/myoutput\",
    dataFormat = \"ukbiobank_csv\",
    extEpochData_timeformat = \"%m/%d/%Y %H:%M:%S\",
    mode = c(1:2),
    do.report = c(2),
   windowsizes = c(5, 900, 3600), # We know that data was stored in 5 second epoch
    desiredtz = \"Europe/London\", # We know that data was collected in the UK
    visualreport = FALSE,
    overwrite = TRUE)

# For externally derived ActiGraph count data in .CSV format assuming
# a study protocol where sensor was not worn during the night:
GGIR(datadir = \"/examplefiles\",
```

```
            outputdir = \"\",
            dataFormat = \"actigraph_csv\",
            mode = 1:5,
            do.report = c(2, 4, 5),
            windowsizes = c(5, 900, 3600),
            threshold.in = round(100 * (5/60), digits = 2),
            threshold.mod = round(2500 * (5/60), digits = 2),
            threshold.vig = round(10000 * (5/60), digits = 2),
            extEpochData_timeformat = \"%m/%d/%Y %H:%M:%S\",
            do.neishabouricounts = TRUE,
            acc.metric = \"NeishabouriCount_x\",
            HASPT.algo = \"NotWorn\",
            HASIB.algo = \"NotWorn\",
            do.visual = TRUE,
            includedaycrit = 10,
            includenightcrit = 10,
            visualreport = FALSE,
            outliers.only = FALSE,
            save_ms5rawlevels = TRUE,
            ignorenonwear = FALSE,
            HASPT.ignore.invalid = FALSE,
            save_ms5raw_without_invalid = FALSE)


          # For externally derived Sensear data in .xls format:
           GGIR(datadir = \"C:/yoursenseweardatafolder\",
               outputdir = \"D:/youroutputfolder\",
               mode = 1:5,
               windowsizes = c(60, 900, 3600),
               threshold.in = 1.5,
               threshold.mod = 3,
               threshold.vig = 6,
               dataFormat = \"sensewear_xls\",
               extEpochData_timeformat = \"%d-%b-%Y %H:%M:%S\",
               HASPT.algo = \"NotWorn\",
               desiredtz = \"America/New_York\",
               overwrite = TRUE,
               do.report = c(2, 4, 5),
               visualreport = FALSE)


    ## End(Not run)
```

`inspect_binFile_brand` *Identify the Device Brand from a Binary File*

Description

This function identifies the brand of a device (e.g., GENEActiv or MATRIX) based on the contents of a binary file. If the file is unrecognized, it returns a default value indicating the device is not recognized.

Usage

```
inspect_binFile_brand(filename)
```

Arguments

filename	The path to the binary file to be inspected.

Details

The function performs the following steps:

1. Reads the file header to check for the presence of a "Device Type" field.
2. If the device type contains the string "GENEActiv", the device is identified as a GENEActiv device.
3. If the "Device Type" field is not found, it checks for a MATRIX device by validating a specific header value ("MDTC").
4. If neither device type is detected, the function returns "not_recognised".

Value

A character string representing the detected device brand:

- 2: Indicates the device is a GENEActiv device.
- 7: Indicates the device is a MATRIX device.
- "not_recognised": Indicates the file's brand is not recognized.

Author(s)

Jairo H Migueles <jairo@jhmigueles.com>

---

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)
```

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)
```

load_params *Load default parameters*

Description

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

Usage

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

Arguments

topic	Character vector with parameter groups to be loaded.

Value

Lists of parameter objects

Author(s)

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

part6AlignIndividuals *part6AlignIndividuals*

Description

Align individual time series per household where households are identified by the character or number string between the first and second '-' in the filename.

Usage

```
part6AlignIndividuals(GGIR_ts_dir = NULL, outputdir = NULL,
                      path_ggirms = NULL, desiredtz = "", verbose = TRUE)
```

Arguments

GGIR_ts_dir	Character, path to time series directory in the GGIR output
outputdir	Directory where you would like to store the output
path_ggirms	path to GGIR created folder named meta, with the milestone data files
desiredtz	Character, specifying the timezone database name of the timezone the data was collected in.
verbose	See details in GGIR.

Value

no object is returned, only files are created in the output directory

part6PairwiseAggregation *part6PairwiseAggregation*

Description

Pairwise aggregation of the time series of a group.

Usage

```
part6PairwiseAggregation(outputdir = NULL, desiredtz = "", verbose = TRUE)
```

Arguments

outputdir	Directory where you would like to store your results
desiredtz	Character, specifying the timezone database name of the timezone the data was collected in
verbose	See details in GGIR.

Value

No object is returned, only files are created in the output directory

---

`POSIXtime2iso8601` *Convert POSIX to iso8601 timestamp*

Description

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

Usage

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

Arguments

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

Author(s)

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

Examples

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

## End(Not run)
```

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

Description

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

Usage

```
read.myacc.csv(rmc.file=c(), rmc.nrow=Inf, rmc.skip = c(), rmc.dec=".",
                          rmc.firstrow.acc = c(), 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 = NULL,
                          rmc.configtz = NULL,
                          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,
                          rmc.scalefactor.acc = 1,
                          interpolationType=1,
                          PreviousLastValue = c(0, 0, 1),
                          PreviousLastTime = NULL,
                          desiredtz = NULL,
                          configtz = NULL,
                          header = NULL)
```

Arguments

`rmc.file`	Filename of file to be read if it is in the working directory, or full path to the file otherwise.
`rmc.nrow`	Number of rows to read, same as nrow argument in read.csv and nrows in fread. The whole file is read by default (i.e., rmc.nrow = Inf).
`rmc.skip`	Number of rows to skip, same as skip argument in read.csv and in fread.
`rmc.dec`	Decimal used for numbers, same as skip argument in read.csv and in fread.
`rmc.firstrow.acc`	First row (number) of the acceleration data.
`rmc.firstrow.header`	First row (number) of the header. Leave blank if the file does not have a header.
`rmc.header.length`	If file has header, specify header length (numeric).
`rmc.col.acc`	Vector with three column (numbers) in which the acceleration signals are stored

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

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

`rmc.unit.acc` Character with unit of acceleration values: "g", "mg", or "bit"

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

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

`rmc.format.time`
Character string giving a date-time format as used by strptime. Only used for rmc.unit.time: character and POSIX.

`rmc.bitrate` Numeric: If unit of acceleration is a bit then provide bit rate, e.g. 12 bit.

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

`rmc.unsignedbit`
Boolean, if unsignedbit = TRUE means that bits are only positive numbers. if unsignedbit = FALSE then bits are both positive and negative.

`rmc.origin` Origin of time when unit of time is UNIXsec, e.g. 1970-1-1

`rmc.desiredtz` Deprecated, please see `desiredtz`.

`rmc.configtz` Deprecated, please see `configtz`.

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

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

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

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

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

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

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

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

`rmc.scalefactor.acc`
Numeric value (default 1) to scale the acceleration signals via multiplication. For example, if data is provided in m/s2 then by setting this to 1/9.81 we would derive gravitational units.

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

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

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

`desiredtz` Timezone in which device was worn.

`configtz` Timezone in which device was configured. If equal to desiredtz you can leave this in its default value.

`header` Header information that was extracted the previous time this file was read, to be re-used instead of being extracted again.

Details

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

Value

List with objects data holding the time series of acceleration with among others a column named "time" that holds the time expressed in seconds since 1-1-1970, and header if a header was present in the input file.

Author(s)

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

Examples

```
# create test files: No header, with temperature, with time
N = 30
sf = 30
x = Sys.time()+((0:(N-1))/sf)
timestamps = as.POSIXlt(x, 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(),
                    desiredtz = "",
                    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)
```

SSP *Estimated self-similarity parameter*

Description

This function estimates the overall self-similarity parameter (SSP), also known as the scaling exponent or alpha, as well as the piecewise short-term (alpha_1) and long-term (alpha_2) scaling exponents.

Usage

```
SSP(data, scale = 2^(1/8), box_size = 4, m = 1, epochSize)
```

Arguments

data	Univariate time series (must be a vector or data frame)
scale	Specifies the ratio between successive box sizes (by default scale = 2^(1/8))
box_size	Vector of box sizes (must be used in conjunction with scale = "F")
m	An integer of the polynomial order for the detrending (by default m = 1)
epochSize	The epoch size of the data in seconds. Used to convert box sizes to minutes to apply time boundaries for alpha_1 and alpha_2.

Details

The DFA fluctuation can be computed in a geometric scale or for different choices of boxes sizes. In human motor activity, the scaling behavior exhibits a crossover point around 1.5 to 2 hours. Therefore, alpha_1 captures the short-term temporal correlations regulated by multiple physiological controls, while alpha_2 captures long-term fluctuations that are heavily reliant on the central circadian pacemaker.

Value

A list containing three real numbers (all of them between 0 and 2):

alpha_overall	The overall estimated scaling exponent across all box sizes.
alpha_1	The short-term scaling exponent, calculated for time scales smaller than or equal to 90 minutes.
alpha_2	The long-term scaling exponent, calculated for time scales between 120 and 600 minutes.

Note

It is not possible estimating alpha for multiple time series at once.

Author(s)

Ian Meneghel Danilevicz <ian.meneghel-danilevicz@inserm.fr> Victor Barreto Mesquita <victormesquita40@hotmail.com> Jairo H Migueles <jairo@jhmigueles.com>

References

C.-K. Peng, S.V. Buldyrev, S. Havlin, M. Simons, H.E. Stanley, A.L. Goldberger Phys. Rev. E, 49 (1994), p. 1685 Mesquita, Victor & Filho, Florencio & Rodrigues, Paulo. (2020). Detection of crossover points in detrended fluctuation analysis: An application to EEG signals of patients with epilepsy. Bioinformatics. 10.1093/bioinformatics/btaa955.

Examples

```
 # Estimate self-similarity of a very known time series available on R base: the sunspot.year.
 # Then the spend time with each method is compared.
 ## Not run:
   ssp_results = SSP(sunspot.year)
   ssp_results$alpha_overall
   ssp_results$alpha_1
   ssp_results$alpha_2

## End(Not run)
```

---

visualReport *Generate visualisation of time series produced by part 5.*

Description

Function called by GGIR. Not intended for direct use by user

Usage

```
visualReport(metadatadir = c(),
             f0 = c(), f1 = c(),
             verbose = TRUE,
             part6_threshold_combi = NULL, GGIRversion = NULL,
             params_sleep = NULL,
             params_output = NULL,
             params_general = NULL)
```

Arguments

metadatadir	Directory that holds a folder 'meta' and inside this a folder 'basic' which contains the milestone data produced by g.part1. The folderstructure is normally created by g.part1 and GGIR will recognise what the value of metadatadir is.
f0	File index to start with (default = 1). Index refers to the filenames sorted in alphabetical order
f1	File index to finish with (defaults to number of files available, i.e., f1 = 0)
verbose	See GGIR
part6_threshold_combi	See GGIR
GGIRversion	Character with GGIR version number
params_sleep	See GGIR
params_output	See GGIR
params_general	See GGIR

Value

No values, this function only generates a plot

Author(s)

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

---

write_dashboard_parquet

Write consolidated dashboard Parquet output

Description

Creates a single dashboard-ready Parquet file by combining GGIR summary outputs and attaching nested epoch-level time series by day.

Usage

```
write_dashboard_parquet(
  metadatadir = c(),
  params_output = c(),
  params_general = c(),
  params_phyact = c(),
  verbose = TRUE
)
```

Arguments

metadatadir	Directory that holds a folder 'meta' and inside this a folder 'ms5.outraw' which contains the milestone data produced by g.part5. The folder structure is normally created by g.part5 and GGIR will recognise what the value of metadatadir is.
params_output	See details in GGIR.
params_general	See details in GGIR.
params_phyact	See details in GGIR.
verbose	See details in GGIR.

Details

The function reads already-generated CSV reports, starts from Part 5 day summary as the base table, and left-joins additional fields from:

- Part 4 night summary
- Part 2 day summary
- Part 2 participant summary
- quality-control report

It then calls `build_epoch_lists_by_day` to construct a nested `epochs` list-column and merges that into the day-level table by `ID` and `calendar_date`.

Column names are cleaned for SQL-friendly use, common columns are type-cast, and Parquet key-value metadata is embedded (variable definitions, thresholds, behavioral-code mapping, acceleration metric, and estimated epoch length).

Output filename is ID-based for single-participant exports and falls back to a multi-participant name otherwise.

The generated Parquet file can be opened directly in the GGIR Web Dashboard (`https://wadpac.github.io/ggir-web-dashboard/`), where processing happens entirely in the browser.

Value

Invisibly returns the written Parquet file path, or `NULL` (invisibly) if export is skipped because required data are missing. The generated Parquet file can be opened directly in the GGIR Web Dashboard: `https://wadpac.github.io/ggir-web-dashboard/`

Author(s)

Samuel Timileyin Afolabi <samuelafolabimails@gmail.com>

See Also

`GGIR`, `write_epoch_parquet`, `build_epoch_table`

Examples

```
## Not run:
write_dashboard_parquet(
  metadatadir = "path/to/output_run/output_test_run",
  params_output = list(save_dashboard_parquet = TRUE),
  params_general = list(desiredtz = "", acc.metric = "ENMO"),
  params_phyact = list(part6_threshold_combi = "40_100_400"),
  verbose = TRUE
)

## End(Not run)
```

Index

∗ **IO**
    write_dashboard_parquet, 71
∗ **datasets**
    data.calibrate, 8
    data.getmeta, 9
    data.inspectfile, 9
    data.metalong, 10
    data.ts, 10

ABI, 5
apply_cosinor_IS_IV_Analyses, 6
applyExtFunction, *15*, *19*, *21*, *23*

build_epoch_table, *72*

cosinor_IS_IV_Analyses, *6*
create_test_acc_csv, 6
create_test_sleeplog_csv, 7

data.calibrate, 8
data.getmeta, 9
data.inspectfile, 9
data.metalong, 10
data.ts, 10
DFA, 11

fread, *41*, *66*
fwrite, *57*

g.analyse, *13*, *21*
g.applymetrics, *43, 44*
g.calibrate, *4*, *8*, 12, *19*, *40, 41*, *67*
g.getbout, 13
g.getmeta, *9*, 14, *17*, *19*
g.impute, *21*, *45*
g.imputeTimegaps, 16
g.inspectfile, *9*, *12*, *15*, 17
g.loadlog, 18
g.part1, *10*, *12*, *17*, *19*, 19, *20–38*, *42, 43*, *68*, *71*
g.part2, *19*, 21, *26*, *30*, *35*, *38*, *45, 46*, *48*, *50*, *53*, *56*
g.part3, 22, *24, 25*, *35*, *38*, *49*, *52*, *56*
g.part4, 24, *26*, *31, 32*, *35*, *38*, *45, 46*, *49*, *56*
g.part5, *10*, 26, *32, 33*, *35–38*, *45, 46*, *48, 49*, *53*, *56, 57*, *72*
g.part6, 27, *34*, *36*, *55*
g.plot5, 28
g.report.part2, 30
g.report.part4, 31
g.report.part5, 32
g.report.part5_dictionary, 33
g.report.part6, 34
g.shell.GGIR, 35
get_nw_clip_block_params, *20*
GGIR, *15*, *18–35*, 35, *42*, *64*, *68*, *70–72*
GGIR-package, 3

HASPT, *50*

inspect_binFile_brand, 61
is.ISO8601, 62
iso8601chartime2POSIX, 63

load_params, 63

part6AlignIndividuals, 64
part6PairwiseAggregation, 64
POSIXtime2iso8601, 65

read.csv, *41*, *66*
read.myacc.csv, *14*, *20*, *42*, 65

SSP, 69
strptime, *42*, *46*, *54*, *67*

visualReport, 70

write_dashboard_parquet, 71
write_epoch_parquet, *72*