

# Package ‘GGIR’

February 2, 2022

**Type** Package

**Title** Raw Accelerometer Data Analysis

**Version** 2.6-0

**Date** 2022-02-02

**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; PLoS ONE 2015). The package has been developed and tested for binary data from 'GENEActiv' <<https://www.activinsights.com/>> and GENEActiv devices (not for sale), .csv-export data from 'Actigraph' <<https://actigraphcorp.com>> devices, and .cwa and .wav-format data from 'Axivity' <<https://axivity.com>>. These devices are currently widely used in research on human daily physical activity. Further, the package can handle accelerometer data file from any other sensor brand providing that the data is stored in csv format and has either no header or a two column header. Also the package allows for external function embedding.

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

**BugReports** <https://github.com/wadpac/GGIR/issues>

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

**Suggests** testthat, covr, knitr, rmarkdown

**Imports** data.table, Rcpp (>= 0.12.10), foreach, doParallel, signal,  
zoo, bitops, matlab, GENEActiv, tuneR, unisensR, ineq,  
read.gt3x, R.utils, activityCounts

**Depends** stats, utils, R (>= 3.5.0)

**NeedsCompilation** yes

**LinkingTo** Rcpp

**VignetteBuilder** knitr

**ByteCompile** yes

**Author** Vincent T van Hees [aut, cre],  
 Zhou Fang [ctb],  
 Jing Hua Zhao [ctb],  
 Joe Heywood [ctb],  
 Evgeny Mirkes [ctb],  
 Severine Sabia [ctb],  
 Joan Capdevila Pujol [ctb],  
 Jairo H Migueles [ctb],  
 Matthew R Patterson [ctb],  
 Dan Jackson [ctb],  
 Lena Kushleyeva [ctb],  
 Mathilde Chen [ctb],  
 Manasa Yerramalla [ctb],  
 Tuomo Nieminen [cph] (holds copyright over read.gt3x code),  
 John Muschelli [cph] (<<https://orcid.org/0000-0001-6469-1750>>, holds  
 copyright over read.gt3x code),  
 Patrick Bos [ctb] (<<https://orcid.org/0000-0002-6033-960X>>)

**Repository** CRAN

**Date/Publication** 2022-02-02 12:20:07 UTC

## R topics documented:

GGIR-package	4
applyExtFunction	6
CalcSleepRegularityIndex	7
chartime2iso8601	8
check_myfun	8
check_params	9
createConfigFile	10
create_test_acc_csv	10
create_test_sleeplog_csv	11
data.calibrate	12
data.getmeta	12
data.inspectfile	13
datadir2fnames	13
extract_params	14
g.abr.day.names	15
g.analyse	15
g.analyse.avy	17
g.analyse.perday	18
g.analyse.perfile	20
g.applymetrics	22
g.binread	23
g.calibrate	24
g.conv.actlog	25
g.convert.part2.long	26
g.create.sp.mat	27

<code>g.createcoordinates</code>	27
<code>g.cwared</code>	28
<code>g.deteomidnight</code>	29
<code>g.dotorcomma</code>	30
<code>g.downsample</code>	31
<code>g.extractheadervars</code>	31
<code>g.fragmentation</code>	32
<code>g.getbout</code>	34
<code>g.getidfromheaderobject</code>	35
<code>g.getM5L5</code>	36
<code>g.getmeta</code>	37
<code>g.getstarttime</code>	39
<code>g.impute</code>	39
<code>g.imputeTimegaps</code>	41
<code>g.inspectfile</code>	41
<code>g.intensitygradient</code>	42
<code>g.IVIS</code>	43
<code>g.loadlog</code>	44
<code>g.part1</code>	45
<code>g.part2</code>	51
<code>g.part3</code>	56
<code>g.part4</code>	59
<code>g.part4_extractid</code>	60
<code>g.part5</code>	61
<code>g.part5.addfirstwake</code>	62
<code>g.part5.addsib</code>	63
<code>g.part5.classifyNaps</code>	64
<code>g.part5.definedays</code>	65
<code>g.part5.fixmissingnight</code>	66
<code>g.part5.handle_lux_extremes</code>	66
<code>g.part5.lux_persegment</code>	67
<code>g.part5.onsetwaketiming</code>	68
<code>g.part5.savetimeseries</code>	68
<code>g.part5.wakesleepwindows</code>	69
<code>g.plot</code>	70
<code>g.plot5</code>	71
<code>g.readaccfile</code>	72
<code>g.readtemp_movisens</code>	74
<code>g.report.part2</code>	75
<code>g.report.part4</code>	75
<code>g.report.part5</code>	77
<code>g.shell.GGIR</code>	78
<code>g.sib.det</code>	81
<code>g.sib.plot</code>	83
<code>g.sib.sum</code>	83
<code>g.sibreport</code>	84
<code>g.wavread</code>	85
<code>g.weardec</code>	85

getFirstTimestamp . . . . .	86
getfolderstructure . . . . .	87
getStartEnd . . . . .	87
getStartEndNumeric . . . . .	88
get_nw_clip_block_params . . . . .	89
get_starttime_weekday_meantemp_truncdata . . . . .	90
HASIB . . . . .	91
HASPT . . . . .	92
identify_levels . . . . .	93
is.ISO8601 . . . . .	94
isfilelist . . . . .	94
ismovisens . . . . .	95
iso8601chartime2POSIX . . . . .	95
is_this_a_dst_night . . . . .	96
load_params . . . . .	97
numUnpack . . . . .	97
parseGT3Xggir . . . . .	98
POSIXtime2iso8601 . . . . .	99
read.gt3x_ggir . . . . .	99
read.myacc.csv . . . . .	101
resample . . . . .	103
updateBlocksize . . . . .	104

<b>Index</b>	<b>105</b>
--------------	------------

---

GGIR-package	<i>A package to process multi-day raw accelerometer data</i>
--------------	--

---

## Description

Disclaimer: If you are a new GGIR user then please see [package vignette](#) for an introduction to GGIR.

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

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

You can thank us for sharing the code in this package and for developing it as a generic purpose tool by citing the package name and by citing the supporting publications (e.g. Migueles et al. 2019) in your publications.

## Details

Package:	GGIR
Type:	Package
Version:	2.6-0
Date:	2022-02-02

License: LGPL ( $\geq 2.0$ ,  $< 3$ )  
Discussion group: <https://groups.google.com/forum/#!forum/rpackageggir>

### Author(s)

- Vincent T van Hees <[v.vanhees@accelting.com](mailto:v.vanhees@accelting.com)> main creator and developer
- Zhou Fang developed calibration algorithm used in function [g.calibrate](#)
- Jing Hua Zhao <[jinghua.zhao@mrc-epid.cam.ac.uk](mailto:jinghua.zhao@mrc-epid.cam.ac.uk)> co-developed function [g.binread](#)
- Joe Heywood helped develop the functionality to process specific recording days
- Evgeny Mirkes created function [g.cwaread](#)
- 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 <[jairoh@ugr.es](mailto:jairoh@ugr.es)> helped to improve various functions
- Dan Jackson helped with unpack function for AX3 data.
- Matthew R Patterson helped with enhancing the visual report.
- Lena Kushleyeva helped fix bug in sleep detection.

### References

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

### Examples

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

```

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

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

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

```

---

applyExtFunction      *Apply external function to acceleration data.*

---

## Description

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

## Usage

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

## Arguments

data	Data data.frame as present internally in <a href="#">g.getmeta</a> . It has at least four columns of which the first is the timestamp followed by the x, y, and z acceleration.
myfun	See details, in short: myfun is a list object that holds the external function to be applied to the data and various parameters to aid in the process.
sf	Sample frequency (Hertz) of the data object
ws3	Short epoch size (first value of windowsizes in <a href="#">g.getmeta</a> ).
interpolationType	See <a href="#">g.getmeta</a>

## Details

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

**Value**

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

**Author(s)**

Vincent T van Hees <[v.vanhees@accelting.com](mailto:v.vanhees@accelting.com)>

---

CalcSleepRegularityIndex

*Calculates Sleep Regularity Index*

---

**Description**

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

**Usage**

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

**Arguments**

data	Data.frame produced by function <a href="#">g.sib.det</a> .
epochsize	Numeric value of epoch size in seconds.
desiredtz	Character with timezone database name, see also <a href="#">g.getmeta</a>

**Details**

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

**Value**

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

**Author(s)**

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

**References**

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

---

chartime2iso8601      *Convert character timestamps to iso8601 timestamp*

---

**Description**

To avoid ambiguities when sharing and comparing timestamps. All timestamps are expressed in iso8601 format: [https://en.wikipedia.org/wiki/ISO\\_8601](https://en.wikipedia.org/wiki/ISO_8601)

**Usage**

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

**Arguments**

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

**Examples**

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

---

check\_myfun      *Checks myfun object before it is passed to applyExtfunction*

---

**Description**

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

**Usage**

```
check_myfun(myfun, window sizes)
```



**Arguments**

myfun            See [applyExtFunction](#)  
 window sizes    See [g.getmeta](#)).

**Value**

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

**Author(s)**

Vincent T van Hees <[v.vanhees@accelting.com](mailto:v.vanhees@accelting.com)>

---

check_params	<i>Check default parameters</i>
--------------	---------------------------------

---

**Description**

Checks parameter objects for class and logical combinations. Called from [extract\\_params](#). Not intended for direct use by GGIR users.

**Usage**

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

**Arguments**

params\_sleep    List with sleep parameters  
 params\_metrics    List with parameters related to metrics  
 params\_rawdata    List with parameters related to raw data reading and processing  
 params\_247        List with parameters related to 24/7 behavioural analysis, which includes anything that does not fit with physical activity or sleep research  
 params\_phyact    List with parameters related to physical activity analysis  
 params\_cleaning    List with parameters related to cleaning the time series, including masking and imputation  
 params\_output    List with parameters related to how GGIR stores its output  
 params\_general    List with parameters related to general topics

**Value**

Lists of updated parameter objects

**Author(s)**

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

---

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

---

**Description**

Only used inside [g.shell.GGIR](#). Not intended for direct use by user.

**Usage**

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

**Arguments**

config.parameters

List with all arguments used in [g.shell.GGIR](#).

**Author(s)**

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

---

create\_test\_acc\_csv      *Creates csv data file for testing purposes*

---

**Description**

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

**Usage**

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

**Arguments**

sf                      Sample frequency in Hertz, the default here is low to minimize file size

Nmin                    Number of minutes (minimum is 2000)

storagelocation

Location where the test file named testfile.csv will be stored If no value is provided then the function uses the current working directory

**Value**

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

**Examples**

```
## Not run:
  create_test_acc_csv()

## End(Not run)
```

---

```
create_test_sleeplog_csv
```

*Creates csv sleeplog file for testing purposes*

---

**Description**

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

**Usage**

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

**Arguments**

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

**Value**

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

**Examples**

```
## Not run:
  create_test_sleeplog_csv()

## End(Not run)
```

---

data.calibrate	<i>Example output from g.calibrate</i>
----------------	--

---

**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	<i>Example output from g.getmeta</i>
--------------	--------------------------------------

---

**Description**

data.getmeta is example output from [g.getmeta](#)

**Usage**

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

**Format**

The format is: chr "data.getmeta"

**Source**

The data was collected on one individual for testing purposes

**Examples**

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

---

data.inspectfile      *Example output from g.inspectfile*

---

**Description**

data.inspectfile is example output from [g.inspectfile](#)

**Usage**

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

**Format**

The format is: chr "data.inspectfile"

**Source**

The data was collected on one individual for testing purposes

**Examples**

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

---

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

---

**Description**

Uses input argument datadir from [g.part1](#) and the output from [isfilelist](#) to generate vector of filenames

**Usage**

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

**Arguments**

datadir	See <a href="#">g.part1</a>
filelist	Produced by <a href="#">isfilelist</a>

**Value**

Character vector of filenames

**Author(s)**

Vincent T van Hees <[v.vanhees@accelting.com](mailto:v.vanhees@accelting.com)>

**Examples**

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

## End(Not run)
```

---

extract_params	<i>Extract parameters from input and add them to params</i>
----------------	---

---

**Description**

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

**Usage**

```
extract_params(params_sleep = c(), params_metrics = c(),
              params_rawdata = c(), params_247 = c(),
              params_phyact = c(), params_cleaning = c(),
              params_output = c(), params_general = c(), input = c(),
              configfile_csv = c())
```

**Arguments**

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

**Value**

Lists of updated parameter objects

**Author(s)**

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

---

g.abr.day.names	<i>Abbreviates daynames to numbers, needed for report generation in <a href="#">g.plot5</a></i>
-----------------	---

---

**Description**

Abbreviates daynames Monday becomes MON and Sunday becomes SUN

**Usage**

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

**Arguments**

daynames            Vector of daynames in character format

**Author(s)**

Vincent T van Hees <[v.vanhees@accelting.com](mailto:v.vanhees@accelting.com)>

**Examples**

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

---

g.analyse	<i>Function to analyse meta-data generated by <a href="#">g.getmeta</a> and <a href="#">g.impute</a></i>
-----------	--

---

**Description**

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

**Usage**

```
g.analyse(I, C, M, IMP, params_247 = c(), params_phyact = c(),
          quantiletype = 7, includedaycrit = 16,
          idloc = 1, snloc = 1, selectdaysfile=c(),
          dayborder=0, desiredtz = "", myfun=c(), ...)
```

**Arguments**

I	the output from function <a href="#">g.inspectfile</a>
C	the output from function <a href="#">g.calibrate</a>
M	the output from function <a href="#">g.getmeta</a>
IMP	the output from function <a href="#">g.impute</a>
params_247	See <a href="#">g.part2</a>
params_phyact	See <a href="#">g.part2</a>
quantiletype	type of quantile function to use (default recommended). For details, see quantile function in STATS package
includedaycrit	See <a href="#">g.part1</a>
idloc	See <a href="#">g.part1</a>
snloc	If value = 1 (default) the code assumes that device serial number is stored in the obvious header field. If value = 2 the code uses the character string between the first and second character '_' in the filename as the serial number
selectdaysfile	See <a href="#">g.part1</a>
dayborder	See <a href="#">g.part1</a>
desiredtz	See <a href="#">g.part1</a>
myfun	External function object to be applied to raw data, see <a href="#">g.getmeta</a> .
...	Any argument used in the previous version of g.analyse, which will now be used to overrule the arguments specified with the parameter objects.

**Value**

g.analyse	generated two data.frames
summary	summary for the file that was analysed
daysummary	summary per day for the file that was analysed

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

**Author(s)**

Vincent T van Hees <[v.vanhees@accelting.com](mailto:v.vanhees@accelting.com)>

**Examples**

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



```

#autocalibration:
C = g.calibrate(datafile)

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

## End(Not run)
#impute meta-data:
IMP = g.impute(M = data.getmeta, I = data.inspectfile)

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

```

---

g.analyse.avy

*Function supports [g.analyse](#). Not intended for direct use by user.*


---

## Description

Generates average day analyses and fills corresponding output matrix, [g.analyse](#).

## Usage

```

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

```

## Arguments

doquan	Boolean whether quantile analysis should be done
averageday	As produced by <a href="#">g.impute</a>
M	As produced by <a href="#">g.getmeta</a>
IMP	As produced by <a href="#">g.impute</a>
t_TWDI	Same as qwindow as described in <a href="#">g.analyse</a>
quantiletype	see <a href="#">g.analyse</a>
ws3	Epoch size in seconds
doiglevels	Boolean to indicate whether iglevels should be calculated
firstmidnighti	see <a href="#">g.deteomidnight</a>
ws2	see <a href="#">g.weardec</a>
midnightsi	see <a href="#">g.deteomidnight</a>
params_247	See <a href="#">g.part2</a>
...	Any argument used in the previous version of <a href="#">g.analyse.avday</a> , which will now be used to overrule the arguments specified with the parameter objects.

**Value**

InterdailyStability

IntradailyVariability

igfullr\_names

igfullr

QUAN

qlevels\_names

ML5AD

ML5AD\_names

**Author(s)**

Vincent T van Hees &lt;v.vanhees@accelting.com&gt;

---

`g.analyse.perday`*Function supports [g.analyse](#). Not intended for direct use by user.*

---

**Description**Generates day specific analyses and fills corresponding output matrix, [g.analyse](#).**Usage**

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

**Arguments**`selectdaysfile` see [g.analyse](#)`ndays` Number of days in file`firstmidnighti` see [g.detecmidnight](#)`time` timestamp column from metalong converted to character`nfeatures` estimate of number of variables that need to be stored in the output matrix`midnightsi` see [g.detecmidnight](#)

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

**Value**

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

**Author(s)**

Vincent T van Hees <[v.vanhees@accelting.com](mailto:v.vanhees@accelting.com)>

---

g.analyse.perfile      *Function supports [g.analyse](#). Not intended for direct use by user.*

---

## Description

Generates recording specific analyses and fills corresponding output matrix, [g.analyse](#).

## Usage

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

## Arguments

ID	Person Identification number, this can be numeric or character
fname	see <a href="#">g.analyse.perday</a>
deviceSerialNumber	As produced by <a href="#">g.extractheadervars</a>
BodyLocation	as produced by <a href="#">g.extractheadervars</a>
startt	First timestamp in metalong
I	output <a href="#">g.inspectfile</a>
LC2	see <a href="#">g.impute</a>
LD	length data in minutes
dcomplscore	see <a href="#">g.impute</a>
LMp	length measurement based on study protocol (minutes)
LWp	length of sensor worn based on study protocol (minutes)
C	output <a href="#">g.calibrate</a>
lookat	indices of metashort column to analyse
AveAccAve24hr	Average acceleration in an average 24 hour cycle
colnames_to_lookat	Names of columns to look at, corresponding to argument lookat
QUAN	Results quantile analysis on the average day produced by <a href="#">g.analyse.avday</a>
ML5AD	Results ML5 analyses on the average day produced by <a href="#">g.analyse.avday</a>
ML5AD_names	Columns names corresponding to ML5AD

igfullr	Results intensity gradient (ig) analysis on the average day produced by <a href="#">g.analyse.avday</a>
igfullr_names	Columns names corresponding to igfullr
daysummary	object produced by <a href="#">g.analyse.perday</a>
ds_names	column names corresponding to daysummary
includedaycrit	see <a href="#">g.analyse</a>
strategy	see <a href="#">g.analyse</a>
hrs.del.start	see <a href="#">g.analyse</a>
hrs.del.end	see <a href="#">g.analyse</a>
maxdur	see <a href="#">g.analyse</a>
windowsizes	see <a href="#">g.getmeta</a>
idloc	see <a href="#">g.analyse</a>
snloc	see <a href="#">g.analyse</a>
wdayname	character with weekdayname
doquan	Boolean whether quantile analysis should be done
qllevels_names	object produced by <a href="#">g.analyse.avday</a>
doiglevels	Boolean to indicate whether iglevels should be calculated
tooshort	0 (file not too short) or 1 (file too short)
InterdailyStability	see <a href="#">g.IVIS</a>
IntradailyVariability	see <a href="#">g.IVIS</a>
IVIS_windowsize_minutes	see <a href="#">g.IVIS</a>
qwindow	see <a href="#">g.analyse</a>
longitudinal_axis_id	Index of axis for which the angle correlates most strongly across 24 hours as calculated inside g.analyse. For hip worn accelerometer this helps to check which axis was the vertical axis. The estimate may not be informative for other attachment locations.

**Value**

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

**Author(s)**

Vincent T van Hees <[v.vanhees@accelting.com](mailto:v.vanhees@accelting.com)>

---

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

---

### Description

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

### Usage

```
g.applymetrics(data, sf, ws3, metrics2do,
               n = 4, lb = 0.2, hb = 15)
```

### Arguments

<code>data</code>	Three column matrix with x, y, and z acceleration data
<code>n</code>	filter order, only needed if a metric is selected that involves a frequency filter
<code>sf</code>	sample frequency
<code>ws3</code>	Epoch size in seconds
<code>metrics2do</code>	Dataframe with Boolean indicator for all metrics whether they should be extracted or not. For instance, <code>metrics2do\$do.bfen = TRUE</code> , indicates that the <code>bfen</code> metric should be extracted
<code>lb</code>	Lower boundary of cut-off frequencies
<code>hb</code>	Higher boundary of cut-off frequencies

### Value

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

### Author(s)

Vincent T van Hees <[v.vanhees@accelting.com](mailto:v.vanhees@accelting.com)>

### Examples

```
Gx = runif(n=10000,min=0,max=2)
Gy = runif(n=10000,min=1,max=3)
Gz = runif(n=10000,min=0,max=2)
data = cbind(Gx, Gy, Gz)
metrics2do = data.frame(do.bfen=TRUE,do.enmo=TRUE,do.lfenmo=FALSE,
do.en=FALSE,do.hfen=FALSE,do.hfenplus=FALSE,do.mad=FALSE,do.anglex=FALSE,
do.angley=FALSE,do.anglez=FALSE,do.roll_med_acc_x=FALSE,
do.roll_med_acc_y=FALSE,do.roll_med_acc_z=FALSE,
do.dev_roll_med_acc_x=FALSE,do.dev_roll_med_acc_y=FALSE,
do.dev_roll_med_acc_z=FALSE,do.enmoa=FALSE,
do.lfx=FALSE, do.lfy=FALSE, do.lfz=FALSE,
do.hfx=FALSE, do.hfy=FALSE, do.hfz=FALSE,
do.bfx=FALSE, do.bfy=FALSE, do.bfz=FALSE,
```

```
do.zcx=FALSE, do.zcy=FALSE, do.zcz=FALSE, do.brondcounts=FALSE)
extractedmetrics = g.applymetrics(data,n=4,sf=40,ws3=5,metrics2do)
```

---

g.binread	<i>function to read binary files as produced by the accelerometer named 'Genea', not to be confused with the 'GENEActiv' (see package GENEAread for this)</i>
-----------	---

---

### Description

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

### Usage

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

### Arguments

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

### Details

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

### Value

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

### Author(s)

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

---

g.calibrate	<i>function to estimate calibration error and make recommendation for addressing it</i>
-------------	---

---

### 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 calibration procedure.

### Usage

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

### Arguments

datafile	Name of accelerometer file
params_rawdata	See <a href="#">g.part1</a>
params_general	See <a href="#">g.part1</a>
params_cleaning	See <a href="#">g.part1</a>
...	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 collected with GENEActiv monitor.



**Author(s)**

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

**References**

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

**Examples**

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

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

## End(Not run)
```

---

g.conv.actlog

---

*Function to read activity log and make it useful for the rest of GGIR.*


---

**Description**

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

**Usage**

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

**Arguments**

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

qwindow\_dateformat

Character specifying the date format used in the activity log.

**Value**

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

**Author(s)**

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

---

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

---

**Description**

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

**Usage**

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

**Arguments**

daySUMMARY    Object available inside *g.report.part2*

**Value**

Data.frame with long format version of daySUMMARY

**Author(s)**

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

---

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

---

**Description**

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

**Usage**

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

**Arguments**

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

**Value**

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

**Author(s)**

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

---

g.createcoordinates      *Create coordinates for g.plot*

---

**Description**

Function creates the coordinates for the blocks [g.plot](#) Function not designed for direct use by package user.

**Usage**

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

**Arguments**

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

**Value**

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

**Author(s)**

Vincent van Hees <[v.vanhees@accelting.com](mailto:v.vanhees@accelting.com)>

---

g.cwaread	<i>Function to read .cwa-format files as produced by the accelerometer named 'Axivity'</i>
-----------	--

---

**Description**

For reading .cwa-format data, if you have .wav format data then see function [g.wavread](#)

**Usage**

```
g.cwaread(fileName, start = 0, end = 0, progressBar = FALSE,
  desiredtz = "", configtz = c(), interpolationType=1)
```

**Arguments**

fileName	filename (required)
start	start point for reading data, this can either be a timestamp "year-month-day hr:min:sec" or a page number (optional)
end	end point for reading data, this can either be a timestamp "year-month-day hr:min:sec" or a page number (optional)
progressBar	Is trigger to switch on/off the text progress bar. If progressBar is TRUE then the function displays the progress bar but it works slightly slower
desiredtz	Desired timezone, see documentation <a href="#">g.getmeta</a>
configtz	Only functional for AX3 cwa data at the moment. Timezone in which the accelerometer was configured. Only use this argument if the timezone of configuration and timezone in which recording took place are different.
interpolationType	See <a href="#">g.getmeta</a>

**Value**

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

**Author(s)**

Evgeny Mirkes <em322@leicester.ac.uk> Vincent van Hees <v.vanhees@accelting.com>

---

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

---

**Description**

Detect all midnights in a time series

**Usage**

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

**Arguments**

time	Vector of timestamps, either in iso8601 or in POSIX format
desiredtz	See <a href="#">g.part2</a>
dayborder	see <a href="#">g.analyse</a>

**Value**

Output of the function is list containing the following objects:

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

**Author(s)**

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

---

g.dotorcomma	<i>Assesses whether decimals in fileheader are stored with comma or dot separated decimals</i>
--------------	--

---

### Description

The function is used by [g.readaccfile](#) to assess how numeric data should be interpreted

### Usage

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

### Arguments

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

### Value

Character object showing how decimals are separated

### Author(s)

Vincent T van Hees <[v.vanhees@accelting.com](mailto:v.vanhees@accelting.com)>

### Examples

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

---

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

---

**Description**

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

**Usage**

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

**Arguments**

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

**Value**

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

**Author(s)**

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

**Examples**

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

---

`g.extractheadervars`      *Extracts header variables from header object*

---

**Description**

Function is not intended for direct interaction by package end user

**Usage**

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

**Arguments**

I                      Object produced by [g.inspectfile](#)

**Value**

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

**Author(s)**

Vincent T van Hees <[v.vanhees@accelting.com](mailto:v.vanhees@accelting.com)>

**Examples**

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

---

`g.fragmentation`                      *Fragmentation metrics from time series.*

---

**Description**

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

**Usage**

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

**Arguments**

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



## Details

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

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

- Inactive - if name includes the character strings "day\_IN\_unbt" or "day\_IN\_bts".
- LIPA - If name includes the character strings "day\_LIG\_unbt" or "day\_LIG\_bts".
- MVPA - If name includes the character strings "day\_MOD\_unbt", "day\_VIG\_unbt", or "day\_MVPA\_bts"

## Value

List with Character object showing how decimals are separated

TP_PA2IN	Transition probability physical activity to inactivity
TP_IN2PA	Transition probability physical inactivity to activity
Nfrag_IN2LIPA	Number of inactivity fragments succeeded by LIPA (light physical activity)
TP_IN2LIPA	Transition probability physical inactivity to LIPA
Nfrag_IN2MVPA	Number of inactivity fragments succeeded by MVPA (moderate or vigorous physical activity)
TP_IN2MVPA	Transition probability physical inactivity to MVPA
Nfrag_MVPA	Number of MVPA fragments
Nfrag_LIPA	Number of LIPA fragments
mean_dur_MVPA	mean MVPA fragment duration
mean_dur_LIPA	mean LIPA fragment duration
Nfrag_IN	Number of inactivity fragments
Nfrag_PA	Number of activity fragments
mean_dur_IN	mean duration inactivity fragments
mean_dur_PA	mean duration activity fragments
Gini_dur_IN	Gini index corresponding to inactivity fragment durations
Gini_dur_PA	Gini index corresponding to activity fragment durations
CoV_dur_IN	Coefficient of Variance corresponding to inactivity fragment durations
CoV_dur_PA	Coefficient of Variance corresponding to activity fragment durations
alpha_dur_IN	Alpha of the fitted power distribution through inactivity fragment durations
alpha_dur_PA	Alpha of the fitted power distribution through activity fragment durations
x0.5_dur_IN	x0.5 corresponding to alpha_dur_IN
x0.5_dur_PA	x0.5 corresponding to alpha_dur_PA
W0.5_dur_IN	W0.5 corresponding to alpha_dur_IN
W0.5_dur_PA	W0.5 corresponding to alpha_dur_PA
NFragPM_IN	Number of IN fragments per minutes in IN
NFragPM_PA	Number of PA fragments per minutes in PA
SD_dur_IN	Standard deviation in the duration of inactivity fragments
SD_dur_PA	Standard deviation in the duration of physical activity fragments

**Author(s)**

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

**Examples**

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

## End(Not run)
```

---

g.getbout

*function to calculate bouts from vector of binary classes*

---

**Description**

To detect bouts of behaviour in time series. The function is used by [g.analyse](#)

**Usage**

```
g.getbout(x, boutduration, boutcriter = 0.8, closedbout = FALSE,
          bout.metric=6, ws3=5)
```

**Arguments**

x	vector of zeros and/or ones to be screened for bouts of ones
boutduration	duration of bout in epochs
boutcriter	Minimum percentage of boutduration for which the epoch values are expected to meet the threshold criterium
closedbout	TRUE if you want breaks in bouts to be counted towards time spent in bouts (argument only active for bout.metric 1 and 2)
bout.metric	If value=1 the code uses the MVPA bout definition as has been available since 2014 (see papers by Sabia AJE 2014 and da Silva IJE 2014). Here, the algorithm looks for 10 minute windows in which more than XX percent of the epochs are above mvpathreshold, and then counts the entire window as mvpa. If value=2 the code looks for groups of epochs with a value above mvpathreshold that span a time window of at least mvpadur minutes in which more than boutcriter percent of the epochs are above the threshold. The motivation for the defition 1 was: A person who spends 10 minutes in MVPA with a 2 minute break in the

middle is equally active as a person who spends 8 minutes in MVPA without taking a break. Therefore, both should be counted equal and counted as 10 minute MVPA bout. The motivation for the definition 2 is: not counting breaks towards MVPA may simplify interpretation and still counts the two persons in the example as each others equal. If value=3, using sliding window across the data to test bout criteria per window and do not allow for breaks larger than 1 minute and with fraction of time larger than the boutcriter threshold. If value=4, same as 3 but also requires the first and last epoch to meet the threshold criteria. If value=5, same as 4, but now looks for breaks larger than a minute such that 1 minute breaks are allowed, and the fraction of time that meets the threshold should be equal than or greater than the bout.criter threshold. If value=6, algorithm improved (2021) to check for first and last epoch.

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

### Value

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

### Author(s)

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

### Examples

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

---

g.getidfromheaderobject

*Extracts participant identifier from header object*

---

### Description

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

### Usage

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

### Arguments

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

**Value**

Participant identifier as character

**Author(s)**

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

**Examples**

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

## End(Not run)
```

---

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

---

**Description**

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

**Usage**

```
g.getM5L5(varnum,ws3,t0_LFMF,t1_LFMF,M5L5res,winhr,qM5L5=c(),
  iglevels=c(), MX.ig.min.dur=10)
```

**Arguments**

<code>varnum</code>	Numeric vector of epoch values
<code>ws3</code>	Small epoch size in seconds
<code>t0_LFMF</code>	Start hour of the day for the M5L5 analyses, e.g. 0 for midnight
<code>t1_LFMF</code>	End hour of the day for the M5L5 analyses, e.g. 24 for midnight
<code>M5L5res</code>	Resolution of the M5L5 analyses in minutes
<code>winhr</code>	window size of M5L5 analyses, e.g. 5 hours
<code>qM5L5</code>	Percentiles (quantiles) to be calculated over L5 and M5 window.
<code>iglevels</code>	See <a href="#">g.analyse</a> . If provided then the intensity gradient will be calculated for all MX windows larger or equal than argument <code>MX.ig.min.dur</code>
<code>MX.ig.min.dur</code>	Minimum MX duration needed in order for intensity gradient to be calculated

**Value**

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

**Author(s)**

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

**Examples**

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

---

g.getmeta	<i>Function to extract meta-data (features) from data in accelerometer file</i>
-----------	---

---

**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 GENEActiv and GENEActiv. Further, the function should be able to cope with csv-format data procuded by GENEActiv and Actigraph

**Usage**

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

**Arguments**

datafile	name of accelerometer file
params_metrics	See <a href="#">g.part1</a>
params_rawdata	See <a href="#">g.part1</a>
params_general	See <a href="#">g.part1</a>
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(average(temperate,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)
selectdaysfile	see <a href="#">g.part1</a>
myfun	External function object to be applied to raw data. See details <a href="#">applyExtFunction</a> .
...	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](mailto: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$metashort[1:5,])

## End(Not run)
```

---

`g.getstarttime`      *Extract start time of a measurement*

---

### Description

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

### Usage

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

### Arguments

<code>datafile</code>	Full path to data file
<code>P</code>	Object extracted with <a href="#">g.readaccfile</a>
<code>header</code>	File header extracted with <a href="#">g.inspectfile</a>
<code>mon</code>	Same as in <a href="#">g.dotorcomma</a>
<code>dformat</code>	Same as in <a href="#">g.dotorcomma</a>
<code>desiredtz</code>	Same as in <a href="#">g.dotorcomma</a>
<code>selectdaysfile</code>	See <a href="#">g.part1</a>

### Value

The starttime

### Author(s)

Vincent T van Hees <[v.vanhees@accelting.com](mailto:v.vanhees@accelting.com)>

---

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

---

### Description

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

### Usage

```
g.impute(M, I, params_cleaning = c(),
desiredtz="", dayborder= 0, TimeSegments2Zero =c(), ...)
```

**Arguments**

M	output from <a href="#">g.getmeta</a>
I	output from <a href="#">g.inspectfile</a>
params_cleaning	See <a href="#">g.part1</a>
desiredtz	See <a href="#">g.part1</a>
dayborder	See <a href="#">g.part1</a>
TimeSegments2Zero	Optional data.frame to specify which time segments need to be ignored for the imputation, and acceleration metrics to be imputed by zeros. The data.frame is expected to contain two columns named windowstart and windowend, with the start- and end time of the time segment in POSIXlt class.
...	Any argument used in the previous version of g.impute, which will now be used to overrule the arguments specified with the parameter objects.

**Value**

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

**Author(s)**

Vincent T van Hees <[v.vanhees@accelting.com](mailto:v.vanhees@accelting.com)>

**Examples**

```
## Not run:
#inspect file:
I = g.inspectfile(datafile)
#autocalibration:
C = g.calibrate(datafile)
#get meta-data:
M = g.getmeta(datafile)

## End(Not run)
data(data.getmeta)
data(data.inspectfile)
#impute meta-data:
IMP = g.impute(M=data.getmeta, I=data.inspectfile)
```



---

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

---

**Description**

Removes all sample with a zero in each of the three axes, and then imputes time gaps by the last recorded value per axis normalised to 1\_g\_

**Usage**

```
g.imputeTimegaps(x, xyzCol, timeCol, sf, k=0.25)
```

**Arguments**

<code>x</code>	Data.frame with raw accelerometer data, and a timestamp column with millisecond resolution.
<code>xyzCol</code>	Columnnames or numbers for the x, y and z column
<code>timeCol</code>	Column name or number for the timestamp column
<code>sf</code>	Sample frequency in Hertz
<code>k</code>	Minimum time gap length to be imputed

**Value**

Data.frame based on input x with timegaps inputed

**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 <a href="#">g.getmeta</a>
params_rawdata	See <a href="#">g.part1</a>
configtz	...
...	Any argument used in the previous version of <code>g.getmeta</code> , 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](mailto:v.vanhees@accelting.com)>

---

`g.intensitygradient`    *Intensity gradient calculation*

---

**Description**

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

**Usage**

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

**Arguments**

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

**Value**

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

**Author(s)**

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

**References**

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

---

g.IVIS	<i>Calculates IV and IS</i>
--------	-----------------------------

---

**Description**

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

**Usage**

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

**Arguments**

Xi	Vector with acceleration values, e.g. ENMO metric.
epochsizesecondsXi	Epoch size of the values in Xi expressed in seconds.
IVIS_epochsize_seconds	This argument has been deprecated.
IVIS_windowsize_minutes	Window size of the Intradaily Variability (IV) and Interdaily Stability (IS) metrics in minutes, needs to be able to add up to 24 hours.
IVIS.activity.metric	Metric used for activity calculation. Value = 1, uses continuous scaled acceleration. Value = 2, tries to collapse acceleration into a binary score of rest versus active to try to simulate the original approach.

**Value**

InterdailyStability  
  
IntradailyVariability

**Author(s)**

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

**References**

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

**Examples**

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

---

g.loadlog

*Load and clean sleeplog information*

---

**Description**

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

**Usage**

```
g.loadlog(loglocation=c(), coln1=c(), colid=c(), nnights=c(),
  sleeplogidnum=TRUE, sleeplogsep=",", meta.sleep.folder = c(),
  desiredtz="")
```

**Arguments**

loglocation	Location of the spreadsheet (csv) with sleep log information. See package vignette for explanation on expected format
coln1	Column number in the sleep log spreadsheet where the onset of the first night starts
colid	Column number in the sleep log spreadsheet in which the participant ID code is stored (default = 1)
nnights	Number of nights for which sleep log information should be available. It assumes that this is constant within a study. If sleep log information is missing for certain nights then leave these blank
sleeplogidnum	Should the participant identifier as stored in the sleeplog be interpreted as a number (TRUE=default) or a character (FALSE)?
sleeplogsep	Value used as sep argument for reading sleeplog csv file.
meta.sleep.folder	Path to part3 milestone data, only specify if sleeplog is in advanced format.
desiredtz	See <a href="#">g.part4</a>

**Value**

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

**Author(s)**

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

**Examples**

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

## End(Not run)
```

---

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

---

**Description**

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

**Usage**

```
g.part1(datadir = c(), outputdir = c(), f0 = 1, f1 = c(),
  studyname = c(), myfun = c(), params_metrics = c(), params_rawdata = c(),
  params_cleaning = c(), params_general = c(), ...)
```

**Arguments**

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

studyname	If the datadir is a folder then the study will be given the name of the data directory. If datadir is a list of filenames then the studyname will be used as name for the analysis
myfun	External function object to be applied to raw data. See details <a href="#">applyExtFunction</a> .
params_metrics	See details
params_rawdata	See details
params_cleaning	See details
params_general	See details.
...	If you are working with a non-standard csv formatted files, g.part1 also takes any input arguments needed for function <a href="#">read.myacc.csv</a> and argument rmc.noise from <a href="#">get_nw_clip_block_params</a> . First test these argument with function <a href="#">read.myacc.csv</a> 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 [g.shell.GGIR](#) you have the option to specify 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 [g.shell.GGIR](#). Directly specified argument overrule the configuration file and default values.

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

GGIR part 1 (g.part1) takes the following parameter objects as input:

- params\_metrics:** A list of parameters used to specify the signal metrics that need to be extract in GGIR part 1.
- do.anglex** Boolean, if TRUE calculate metric. For computation specifics see source code of function [g.applymetrics](#)
- do.angley** Boolean, if TRUE calculate metric. For computation specifics see source code of function [g.applymetrics](#)
- do.anglez** Boolean, if TRUE calculate metric. For computation specifics see source code of function [g.applymetrics](#)
- do.zcx** Boolean, if TRUE calculate metric zero-crossing count for x-axis. For computation specifics see source code of function [g.applymetrics](#)
- do.zcy** Boolean, if TRUE calculate metric zero-crossing count for y-axis. For computation specifics see source code of function [g.applymetrics](#)
- do.zcz** Boolean, if TRUE calculate metric zero-crossing count for z-axis. For computation specifics see source code of function [g.applymetrics](#)
- do.enmo** Boolean, if TRUE calculate metric. For computation specifics see source code of function [g.applymetrics](#)
- do.lfenmo** Boolean, if TRUE calculate metric. For computation specifics see source code of function [g.applymetrics](#)

- do.en** Boolean, if TRUE calculate metric. For computation specifics see source code of function [g.applymetrics](#)
- do.mad** Boolean, if TRUE calculate metric. For computation specifics see source code of function [g.applymetrics](#)
- do.enmoa** Boolean, if TRUE calculate metric. For computation specifics see source code of function [g.applymetrics](#)
- do.roll\_med\_acc\_x** Boolean, if TRUE calculate metric. For computation specifics see source code of function [g.applymetrics](#)
- do.roll\_med\_acc\_y** Boolean, if TRUE calculate metric. For computation specifics see source code of function [g.applymetrics](#)
- do.roll\_med\_acc\_z** Boolean, if TRUE calculate metric. For computation specifics see source code of function [g.applymetrics](#)
- do.dev\_roll\_med\_acc\_x** Boolean, if TRUE calculate metric. For computation specifics see source code of function [g.applymetrics](#)
- do.dev\_roll\_med\_acc\_y** Boolean, if TRUE calculate metric. For computation specifics see source code of function [g.applymetrics](#)
- do.dev\_roll\_med\_acc\_z** Boolean, if TRUE calculate metric. For computation specifics see source code of function [g.applymetrics](#)
- do.bfen** Boolean, if TRUE calculate metric. For computation specifics see source code of function [g.applymetrics](#)
- do.hfen** Boolean, if TRUE calculate metric. For computation specifics see source code of function [g.applymetrics](#)
- do.hfenplus** Boolean, if TRUE calculate metric. For computation specifics see source code of function [g.applymetrics](#)
- do.lfen** Boolean, if TRUE calculate metric. For computation specifics see source code of function [g.applymetrics](#)
- do.lfx** Boolean, if TRUE calculate metric. For computation specifics see source code of function [g.applymetrics](#)
- do.lfy** Boolean, if TRUE calculate metric. For computation specifics see source code of function [g.applymetrics](#)
- do.lfz** Boolean, if TRUE calculate metric. For computation specifics see source code of function [g.applymetrics](#)
- do.hfx** Boolean, if TRUE calculate metric. For computation specifics see source code of function [g.applymetrics](#)
- do.hfy** Boolean, if TRUE calculate metric. For computation specifics see source code of function [g.applymetrics](#)
- do.hfz** Boolean, if TRUE calculate metric. For computation specifics see source code of function [g.applymetrics](#)
- do.bfx** Boolean, if TRUE calculate metric. For computation specifics see source code of function [g.applymetrics](#)
- do.bfy** Boolean, if TRUE calculate metric. For computation specifics see source code of function [g.applymetrics](#)
- do.bfz** Boolean, if TRUE calculate metric. For computation specifics see source code of function [g.applymetrics](#)

**do.brondcounts** Boolean, if TRUE calculate metric via R package activityCounts. We call them BrondCounts because there are large number of activity counts in the physical activity and sleep research field. By calling them Brond Counts we clarify that these are the counts proposed by Jan Brond and implemented in R by Ruben Brondeel. The Brond Counts are intended to be an imitation of one the counts produced by one of the closed source ActiLife software by ActiGraph.

**lb** Numeric, lower boundary of the frequency filter (in Hertz) as used in the filter-based metrics.

**hb** Numeric, higher boundary of the frequency filter (in Hertz) as used in the filter-based metrics.

**n** Numeric, order of the frequency filter as used in a variety of metrics.

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

**do.cal** Boolean. Whether to apply auto-calibration or not by [g.calibrate](#). Default and recommended setting is TRUE.

**imputeTimegaps** Boolean 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 that is turned on in some studies.

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

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

**printsummary** Boolean. If TRUE will print a summary when done



- chunksize** Numeric. Value between 0.2 and 1 to specify the size of chunks to be loaded as a fraction of a 12 hour period, e.g. 0.5 equals 6 hour chunks. The default is 1 (12 hrs). For machines with less than 4Gb of RAM memory a value below 1 is recommended.
- dynrange** Numeric, provide dynamic range for accelerometer data to overwrite hardcoded 6 g for GENE and 8 g for other brands
- interpolationType** Integer to indicate type of interpolation to be used when resampling time series (mainly relevant for Axivity sensors), 1=linear, 2=nearest neighbour
- all arguments that start with "rnc."** see function [read.myacc.csv](#) and [get\\_nw\\_clip\\_block\\_params](#)
- params\_cleaning:** A list of parameters used across all GGIR parts related to masking or imputing data, abbreviated as 'cleaning'.
- do.imp** Boolean. Whether to impute missing values (e.g. suspected of monitor non-wear) or not by [g.impute](#) in GGIR part2. Default and recommended setting is TRUE
- TimeSegments2ZeroFile** Character. Path to csv-file holding the data.frame used for argument TimeSegments2Zero in function [g.impute](#)
- data\_cleaning\_file** Character. 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 part 5 and 4, respectively. So, this will be done regardless of whether the rest of GGIR thinks those day(s)/night(s) are valid. Column relyonguider\_part4 allows you to specify for which nights part 4 should fully rely on the guider. See also package vignette.
- excludefirstlast.part5** Boolean. If TRUE then the first and last window (waking-waking or midnight-midnight) are ignored in part 5.
- excludefirstlast** Boolean. If TRUE then the first and last night of the measurement are ignored for the sleep assessment (part 4).
- excludefirst.part4** Boolean. If TRUE then the first night of the measurement are ignored for the sleep assessment (part 4).
- excludelast.part4** Boolean. If TRUE then the last night of the measurement are ignored for the sleep assessment.
- includenightcrit** Numeric. Minimum number of valid hours per night (24 hour window between noon and noon), used for sleep assessment (part 4).
- minimum\_MM\_length.part5** Numeric. Minimum length in hours of a MM day to be included in the cleaned part 5 results.
- selectdaysfile** Character, Functionality designed for the London Centre of Longitudinal studies. Csv file holding the relation between device serial numbers and measurement days of interest.
- strategy** Numeric, how to deal with knowledge about study protocol. value = 1 means select data based on hrs.del.start and hrs.del.end. Value = 2 makes that only the data between the first midnight and the last midnight is used for imputation. Value = 3 only selects the most active X days in the file where X is specified by argument ndayswindow. Value = 4 to only use the data after the first midnight. Used in GGIR part 2
- hrs.del.start** Numeric, how many HOURS after start of experiment did wearing of monitor start? Used in GGIR part 2
- hrs.del.end** Numeric, how many HOURS before the end of the experiment did wearing of monitor definitely end? Used in GGIR part 2
- maxdur** Numeric, How many DAYS after start of experiment did experiment definitely stop? (set to zero if unknown = default). Used in GGIR part2

**ndayswindow** Numeric, If strategy is set to 3 then this is the size of the window as a number of days. Used in GGIR part2

**includedaycrit.part5** Numeric. see [g.report.part5](#)

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

**max\_calendar\_days** Numeric, the maximum number of calendar days to include

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

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

**selectdaysfile** Character. Do not use, this is legacy code for one specific data study. Character pointing at a csv file holding the relationship between device serial numbers (first column) and measurement dates of interest (second and third column). The date format should be dd/mm/yyyy. And the first row if the csv file is assumed to have a character variable names, e.g. "serialnumber" "Day1" and "Day2" respectively. Raw data will be extracted and stored in the output directory in a new subfolder named 'raw'.

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

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

**maxNcores** Numeric. Maximum number of cores to use when argument do.parallel is set to true. GGIR by default uses the maximum number of available cores, but this argument allows you to set a lower maximum.

**acc.metric** Boolean. Which one of the metrics do you want to consider to analyze L5. The metric of interest need to be calculated in M.

**part5\_agg2\_60seconds** Boolean. Whether to use aggregate epochs to 60 seconds as part of the part 5 analysis.

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

**desiredtz** Character, desired timezone: see also <http://en.wikipedia.org/wiki/Zone.tab>

**configtz** Character, Only functional for AX3 cwa data at the moment. Timezone in which the accelerometer was configured. Only use this argument if the timezone of configuration and timezone in which recording took place are different.

**sensor.location** Character, see [g.sib.det](#)

**acc.metric** Character, see [g.sib.det](#)

**window sizes** Numeric vector, three values to indicate the lengths of the windows as in `c(window1,window2,window3)`: window1 is the short epoch length in seconds and by default 5 this is the time window over which acceleration and angle metrics are calculated, window2 is the long epoch length in seconds for which non-wear and signal clipping are defined, default 900. However, window3 is the window length of data used for non-wear detection and by default 3600 seconds. So, when window3 is larger than window2 we use overlapping windows, while if window2 equals window3 non-wear periods are assessed by non-overlapping windows. Window2 is expected to be a multitude of 60 seconds.

**idloc** Numeric. If `idloc = 1` (default) the code assumes that ID number is stored in the obvious header field. Note that for ActiGraph data the ID 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 occurrence of a `'_'`, `' '` (space), `'.'` (dot), and `'-'`, 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.

### 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](mailto: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 [g.shell.GGIR](#) to generate reports. The variables in these reports are the same variables as described in [g.analyse](#).

**Usage**

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

**Arguments**

datadir	Directory where the accelerometer files are stored or list, e.g. "C:/mydata" of accelerometer filenames and directories, e.g. c("C:/mydata/myfile1.bin", "C:/mydata/myfile2.bin").
metadatadir	Directory where the output from <a href="#">g.part1</a> was stored
f0	File index to start with (default = 1). Index refers to the filenames sorted in increasing order
f1	File index to finish with (defaults to number of files available)
myfun	External function object to be applied to raw data. See details <a href="#">applyExtFunction</a> .
params_cleaning	See <a href="#">g.part1</a>
params_247	See details
params_phyact	See details
params_output	See details
params_general	See <a href="#">g.part1</a>
...	To ensure compatibility with R scripts written for older GGIR versions, the user can also provide parameters listed in the <code>params_</code> 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 [g.shell.GGIR](#) you have the option to specify 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 [g.shell.GGIR](#). Directly specified argument overrule the configuration file and default values.

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

The parameter objects used by GGIR part 2 (`g.part2`) that are no already discussed in [g.part1](#) are:

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

**storefolderstructure** Boolean. Store folder structure of the accelerometer data.

**do.part3.pdf** Boolean. In `g.part3`: Whether to generate a pdf for part 3 (default is TRUE).

**timewindow** In `g.part5`: Timewindow over which summary statistics are derived. Value can be "MM" (midnight to midnight), "WW" (waking time to waking time), or both c("MM", "WW").

**save\_ms5rawlevels** Boolean, whether to save the time series classification (levels) as a csv files.

**save\_ms5raw\_format** Character string to specify how data should be stored: either "csv" (default) or "RData". Only used if `save_ms5rawlevels=TRUE`.

- save\_ms5raw\_without\_invalid** Boolean to indicate whether to remove invalid days from the time series output files. Only used if `save_ms5rawlevels=TRUE`.
- epochvalues2csv** Boolean. If `TRUE` then epoch values are exported to a CSV spreadsheet. Here, non-wear time is imputed where possible (default = `FALSE`).
- do.sibreport** Boolean. Applied in `g.part5`. Boolean to indicate whether to generate report for the sustained inactivity bouts (sib).
- do.visual** Boolean. If `g.part4` is run with `do.visual == TRUE` then the function will generate a pdf with a visual representation of the overlap between the sleeplog entries and the accelerometer detections. This can be used to visually verify that the sleeplog entries do not come with obvious mistakes.
- outliers.only** Boolean. Relevant for `do.visual == TRUE`. `Outliers.only == FALSE` will visualise all available nights in the data. `Outliers.only == TRUE` will visualise only for nights with a difference in onset or waking time larger than the variable of argument `criterror`.
- criterror** Numeric. Relevant for `do.visual == TRUE` and `outliers.only == TRUE`. `criterror` specifies the number of minimum number of hours difference between sleep log and accelerometer estimate for the night to be included in the visualisation.
- visualreport** Boolean. If `TRUE` then generate visual report based on combined output from part 2 and 4. This is in beta-version at the moment.
- viewingwindow** Numeric. Centre the day as displayed around noon (value = 1) or around midnight (value = 2).
- week\_weekend\_aggregate.part5** Boolean, see [g.report.part5](#)
- dofirstpage** Boolean, see [g.plot5](#)
- timewindow** Timewindow over which summary statistics are derived. Value can be "MM" (midnight to midnight), "WW" (waking time to waking time), or both `c("MM","WW")`.
- params\_phyact:** A list of parameters related to physical activity as used in `GGIRpart2` and `GGIRpart5`.
- threshold.lig** Numeric. In `g.part5`: Threshold for light physical activity to separate inactivity from light. Value can be one number or an array of multiple numbers, e.g. `threshold.lig =c(30,40)`. If multiple numbers are entered then analysis will be replicated 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. In `g.part5`: Threshold for moderate physical activity to separate light from moderate. Value can be one number or an array of multiple numbers, e.g. `threshold.mod =c(100,110)`. If multiple numbers are entered then analysis will be replicated 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. In `g.part5`: Threshold for vigorous physical activity to separate moderate from vigorous. Value can be one number or an array of multiple numbers, e.g. `threshold.mod =c(400,500)`. If multiple numbers are entered then analysis will be replicated 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.
- closedbout** Boolean, see [g.getbout](#)
- frag.metrics** Character, see [g.fragmentation](#)
- mvpthreshold** Numeric, Acceleration threshold for MVPA estimation in `GGIR part2`. This can be a single number or an array of numbers, e.g. `c(100,120)`. In the later case the code will

estimate MVPA separately for each threshold. If this variable is left blank `c()` then MVPA is not estimated

**boutcritier** Numeric, The variable `boutcritier` is a number between 0 and 1 and defines what fraction of a bout needs to be above the `mvpthreshold`, only used in GGIR part 2

**mvpadur** Numeric, default = `c(1,5,10)`. Three bout duration for which MVPA will be calculated. Only used in GGIR part 2

**bout.metric** Numeric, Specify a metric for bout detection. A description of these bout metrics can be found in the new function [g.getbout](#)

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

**boutdur.in** Numeric, see Durations 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. The default duration values is `c(10,20,30)`, this will start with the identification of 30 minute bouts, followed by 20 minute bouts in the rest of the data, and followed by 10 minute bouts in the rest of the data.

**boutdur.lig** Numeric, see Durations 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. The default duration values is `c(1,5,10)`, this will start with the identification of 10 minute bouts, followed by 5 minute bouts in the rest of the data, and followed by 1 minute bouts in the rest of the data.

**boutcritier.in** Numeric, see A number between 0 and 1 and defines what fraction of a bout needs to be below the light threshold

**boutcritier.lig** Numeric, see A number between 0 and 1 and defines what fraction of a bout needs to be between the light and moderate threshold

**boutcritier.mvpa** Numeric, see A number between 0 and 1 and defines what fraction of a bout needs to be above the `mvpthreshold`

**params\_247:** A list of parameters related to description of 24/7 behaviours that do not fall under conventional physical activity or sleep outcomes, these parameters are used in GGIRpart2 and GGIRpart5:

**qwindow** Numeric or character, To specify windows over which all variables are calculated, e.g. acceleration distribution, number of valid hours, LXX analysis, MVPA. If value = `c(0,24)`, which is the default, all variables will only be calculated over the full 24 hours in a day, If value = `c(8,24)` variables will be calculated over the window 0-8, 8-24 and 0-24. All days in the recording will be segmented based on these values. If you want to use a day specific segmentation then you can set `qwindow` to be the full path to activity diary file. See documentation [g.conv.actlog](#) for details.

**qwindow\_dateformat** Numeric, see [g.conv.actlog](#)

**M5L5res** Numeric, resolution of L5 and M5 analysis in minutes (default: 10 minutes)

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

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

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

- window.summary.size** Numeric, Functionality designed for the London Centre of Longitudinal studies. Size in minutes of the summary window
- iglevels** Numeric, Levels for acceleration value frequency distribution in mg used for intensity gradient calculation (according to the method by Rowlands 2018). By default this is argument is empty and the intensity gradient calculation is not done. The user can either provide a single value (any) to make the intensity gradient use the bins `c(seq(0,4000,by=25),8000)` or the user could specify their own distribution. There is no restriction to the number of levels.
- IVIS\_windowsize\_minutes** Numeric, see [g.IVIS](#)
- IVIS\_epochsize\_seconds** deprecated. Numeric, see [g.IVIS](#)
- IVIS.activity.metric** Numeric, see [g.IVIS](#)
- qM5L5** Numeric, see [g.getM5L5](#)
- MX.ig.min.dur** Numeric, see [g.getM5L5](#)
- LUXthresholds** Numeric. Vector with numeric sequence corresponding to the thresholds used to calculate time spent in LUX ranges.
- LUX\_cal\_constant** Numeric, if both `LUX_cal_constant` and `LUX_cal_exponent` are provided LUX values are converted based on formula  $y = \text{constant} * \exp(x * \text{exponent})$
- LUX\_cal\_exponent** Numeric, if both `LUX_cal_constant` and `LUX_cal_exponent` are provided LUX values are converted based on formula  $y = \text{constant} * \exp(x * \text{exponent})$
- LUX\_day\_segments** Numeric vector with hours at which the day should be segmented for the LUX analysis.
- L5M5window** Argument deprecated after version 1.5-24. This argument used to define the start and end time, in 24 hour clock hours, over which L5M5 needs to be calculated. Now this is done with argument `qwindow`

## 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](mailto: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	<i>Detection of sustained inactivity periods as needed for sleep detection in g.part4.</i>
---------	--

---

## Description

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

## Usage

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

## Arguments

metadataadir	Directory that holds a folder 'meta' and inside this a folder 'basic' which contains the milestone data produced by <a href="#">g.part1</a> . The folderstructure is normally created by g.part1 and g.shell.GGIR will recognise what the value of metadataadir is.
f0	File index to start with (default = 1). Index refers to the filenames sorted in increasing order
f1	File index to finish with (defaults to number of files available)
myfun	External function object to be applied to raw data. See details <a href="#">applyExtFunction</a> .
params_sleep	See details
params_metrics	List, see <a href="#">g.part1</a> .
params_output	List, see <a href="#">g.part1</a>
params_general	List, see <a href="#">g.part1</a>
...	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 [g.shell.GGIR](#) you have the option to specify 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 [g.shell.GGIR](#). Directly specified argument overrule the configuration file and default values.



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

The parameter objects used by GGIR part 3 (g.part3) that are not already discussed in [g.part1](#) or [g.part2](#) are:

**params\_sleep:** A list of parameters used to configure the sleep analysis as performed in GGIR part 3 and 4.

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

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

**sleepwindowType** Character to indicate type of sleeplog, default "SPT". Set to "TimeInBed" if sleep log recorded time in bed to enable calculation of sleep latency and sleep efficiency.

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

**loglocation** Character. Path to csv file with sleep log information. See package vignette for how to format this file.

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

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

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

**ignorenonwear** Boolean, see [g.sib.sum](#)

**constrain2range** Boolean, Whether or not to constrain the range of threshold used in the diary free sleep period time window detection

**HASPT.ignore.invalid** Boolean, see [HASPT](#)

**HASPT.algo** Character, character to indicate what heuristic algorithm to use for detecting the SPT window, see [HASPT](#)

**HASIB.algo** Character, character to indicate what heuristic algorithm to use for detecting the sustained inactivity bouts (SIB), see [HASIB](#)

**Sadeh\_axis** Character, character to indicate which axis to use for the Sadeh1994 algorithm, and other algorithms that related on count-based Actigraphy such as Galland2012.

**sleeplogsep** Character, [g.loadlog](#)

**nap\_model** Character, see [g.part5.classifyNaps](#)

- longitudinal\_axis** Integer to indicate which axis is the longitudinal axis. If not provided function will estimate longitudinal axis. Only used when sensor.location="hip" or HASPT.algo="HorAngle".
- anglethreshold** Numeric, Angle threshold (degrees) for sustained inactivity periods detection, default = 5
- timethreshold** Numeric, time threshold (minutes) for sustained inactivity periods detection, default = 5. This can be specified as multiple thresholds, each of which will be implemented. For example, timethreshold = c(5,10)
- possible\_nap\_window** Numeric, see [g.part5.classifyNaps](#)
- possible\_nap\_dur** Numeric, see [g.part5.classifyNaps](#)

### 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 explanation).
- 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](mailto: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 description of the sleep detection.

### Usage

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

### Arguments

datadir	Directory where the accelerometer files are stored or list of accelerometer file-names and directories
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.part3. When using g.part4 via g.shell.GGIR then g.shell.GGIR will automatically recognise what the value of metadatadir is, so the user does not need to specify this.
f0	File index to start with (default = 1). Index refers to the filenames sorted in increasing order
f1	File index to finish with (defaults to number of files available)
params_sleep	List of parameters used for sleep analysis (GGIR part 3, 4, and 5): see documentation <a href="#">g.part3</a> .
params_metrics	List of parameters used for metrics extraction (GGIR part 1): see documentation <a href="#">g.part1</a> .
params_cleaning	List, see <a href="#">g.part1</a>
params_output	List, see <a href="#">g.part2</a>
params_general	List, see <a href="#">g.part1</a>
...	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 includes a dataframe named nightsummary. This dataframe is used in g.report.part4 to create two reports one per night and one per person. See package vignette paragraph "Output part 4" for description of all the variables.

**Author(s)**

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

**References**

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

**Examples**

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

## End(Not run)
```

---

`g.part4_extractid`      *Extracts ID from filename and finds matching rows in sleeplog*

---

**Description**

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

**Usage**

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

**Arguments**

<code>idloc</code>	See <a href="#">g.part4</a>
<code>fname</code>	Full path to filename
<code>dolog</code>	Boolean to indicate whether to rely on a sleeplog
<code>sleeplogidnum</code>	Should the participant identifier as stored in the sleeplog be interpreted as a number (TRUE=default) or a character (FALSE)?
<code>sleeplog</code>	Sleeplog data.frame passed on from g.part4

**Value**

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

**Author(s)**

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

---

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

---

### Description

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

### Usage

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

### Arguments

datadir	Directory where the accelerometer files are stored or list of accelerometer file-names and directories
metadatadir	Directory that holds a folders 'meta' and inside this a folder 'basic' which contains the milestone data produced by g.part1. The folderstructure is normally created by g.part1 and g.shell.GGIR will recognise what the value of metadatadir is.
f0	File index to start with (default = 1). Index refers to the filenames sorted in increasing order
f1	File index to finish with (defaults to number of files available)
params_sleep	List of parameters used for sleep analysis (GGIR part 3, 4, and 5): see documentation <a href="#">g.part3</a> .
params_metrics	List of parameters used for metrics extraction (GGIR part 1): see documentation <a href="#">g.part1</a> .
params_247	List, see <a href="#">g.part2</a>
params_phyact	List, see <a href="#">g.part2</a>
params_cleaning	List, see <a href="#">g.part1</a>
params_output	List, see <a href="#">g.part2</a>
params_general	List, see <a href="#">g.part1</a>
...	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 includes a dataframe named output. This dataframe is used in g.report.part5 to create two reports one per day and one per person. See package vignette paragraph "Output part 5" for description of all the variables.

**Author(s)**

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

**Examples**

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

## End(Not run)
```

---

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

---

**Description**

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

**Usage**

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

**Arguments**

```
ts
summarysleep_tmp2

nightsi
sleeplog
ID
Nepochsinhour
Nts
SPTE_end
ws3new
```

**Value**

Data.frame ts

**Author(s)**

Vincent T van Hees <[v.vanhees@accelting.com](mailto:v.vanhees@accelting.com)>

---

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

---

**Description**

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

**Usage**

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

**Arguments**

ts

ws3

Nts

S2

desiredtz

j

nightsi

**Value**

Data.frame ts

**Author(s)**

Vincent T van Hees <[v.vanhees@accelting.com](mailto:v.vanhees@accelting.com)>

---

g.part5.classifyNaps *Classify Naps from identified sustained inactivity bouts*

---

### Description

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

### Usage

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

### Arguments

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

### Value

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

### Author(s)

Vincent T van Hees <[v.vanhees@accelting.com](mailto:v.vanhees@accelting.com)>



---

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

---

### Description

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

### Usage

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

### Arguments

nightsi  
wi  
indjump  
nightsi\_bu  
ws3new  
qq\_backup  
ts  
Nts  
timewindowi  
Nwindows

### Value

List of qq and qq\_backup

### Author(s)

Vincent T van Hees <[v.vanhees@accelting.com](mailto:v.vanhees@accelting.com)>

g.part5.fixmissingnight

*Fix missing night in part 4 output*

---

### Description

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

### Usage

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

### Arguments

summarysleep\_tmp2

Object produced by [g.part4](#)

sleeplog

ID

### Value

Corrected summarysleep\_tmp2 object.

### Author(s)

Vincent T van Hees <[v.vanhees@accltelting.com](mailto:v.vanhees@accltelting.com)>

---

g.part5.handle\_lux\_extremes

*Check lux values for extremes and imputes or removes them*

---

### Description

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

### Usage

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

### Arguments

lux

Vector with lux values

**Value**

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

**Author(s)**

Vincent T van Hees <[v.vanhees@accelting.com](mailto:v.vanhees@accelting.com)>

---

`g.part5.lux_persegment`

*Extract key lux variables per segment of the data.*

---

**Description**

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

**Usage**

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

**Arguments**

`ts`

`sse`

`LUX_day_segments`

`ws3new`

**Value**

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

**Author(s)**

Vincent T van Hees <[v.vanhees@accelting.com](mailto:v.vanhees@accelting.com)>

g.part5.onsetwaketiming

*Identify wake and sleep period window timing*

---

### **Description**

Not intended for direct use by GGIR users. Labels timing of waking up and sleep onset as part of [g.part5](#).

### **Usage**

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

### **Arguments**

qqq  
ts  
min  
sec  
hour  
timewindowi  
skiponset  
skipwake

### **Value**

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

### **Author(s)**

Vincent T van Hees <[v.vanhees@accelting.com](mailto:v.vanhees@accelting.com)>

---

g.part5.savetimeseries

*Saves oart 5 time series to csv files*

---

### **Description**

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

**Usage**

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

**Arguments**

ts  
 LEVELS  
 desiredtz      See [g.getmeta](#).  
 rawlevels\_fname  
  
 save\_ms5raw\_format      See [g.part5](#)  
 save\_ms5raw\_without\_invalid      See [g.part5](#)  
 DaCleanFile      Content of data\_cleaning\_file as documented in [g.report.part5](#). Only used in this function if save\_ms5rawlevels is TRUE, and it only affects the time series files stored.  
 includedaycrit.part5      See [g.report.part5](#). Only used in this function if save\_ms5rawlevels is TRUE, and it only affects the time series files stored.  
 ID      If data\_cleaning\_file is used then this argument specifies which participant ID the data correspond with.

**Value**

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

**Author(s)**

Vincent T van Hees <[v.vanhees@accelting.com](mailto:v.vanhees@accelting.com)>

---

g.part5.wakesleepwindows

*Label wake and sleeperperiod window*

---

**Description**

Not intended for direct use by GGIR users. Label wake and sleeperperiod window as part of [g.part5](#).

**Usage**

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

**Arguments**

ts                    data.frame with time series  
summarysleep\_tmp2                    cleaned output from part 4  
  
desiredtz  
nightsi  
sleeplog  
ws3new  
Nts  
ID  
Nepochsinhour

**Value**

Object ts

**Author(s)**

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

---

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

---

**Description**

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

**Usage**

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

**Arguments**

IMP                    output from [g.impute](#)  
M                      output from [g.getmeta](#)  
I                       output from [g.inspectfile](#)  
durplot                number of days to plot

**Value**

function only produces a plot, no values

**Author(s)**

Vincent T van Hees <[v.vanhees@accelting.com](mailto:v.vanhees@accelting.com)>

**Examples**

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

#autocalibration:
C = g.calibrate(datafile)

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

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

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

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

**g.plot5**


---

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

---

**Description**

Function called by [g.shell.GGIR](#) to generate report. Not intended for direct use by user

**Usage**

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

**Arguments**

metadatadir	See <a href="#">g.part2</a>
dofirstpage	Boolean to indicate whether a first page with histograms summarizing the whole measurement should be added
viewingwindow	See <a href="#">g.shell.GGIR</a>
f0	See <a href="#">g.shell.GGIR</a>
f1	See <a href="#">g.shell.GGIR</a>
overwrite	See <a href="#">g.shell.GGIR</a>
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 <a href="#">g.part1</a> )
desiredtz	See <a href="#">g.getmeta</a>
threshold.lig	See <a href="#">g.part5</a>
threshold.mod	See <a href="#">g.part5</a>
threshold.vig	See <a href="#">g.part5</a>

**Value**

No values, this function only generates a plot

**Author(s)**

Vincent T van Hees <[v.vanhees@accelting.com](mailto:v.vanhees@accelting.com)> Matthew R Patterson <[mpatterson@shimmersensing.com](mailto:mpatterson@shimmersensing.com)>

**Examples**

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

## End(Not run)
```

---

g.readaccfile

*Generic function to read large blocks of accelerometer data*


---

**Description**

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

**Usage**

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



**Arguments**

filename	filename
blocksize	Size of blocks (in file pages) to be read
blocknumber	Block number relative to start of file
selectdaysfile	See documentation <a href="#">g.getmeta</a>
filequality	Single row dataframe with columns: filetooshort, filecorrupt, and filedoesnothold-day. All with the value TRUE or FALSE
decn	Character with a dot or a comma, used for interpreting samplefrequency in the file header. decn is derived with <a href="#">g.dotorcomma</a>
ws	Larger window size for non-detection, see documentation <a href="#">g.part2</a>
PreviousEndPage	Page number on which previous block ended (automatically assigned within <a href="#">g.getmeta</a> and <a href="#">g.calibrate</a> ).
inspectfileobject	Output from the function <a href="#">g.inspectfile</a> .
params_rawdata	See <a href="#">g.part1</a>
params_general	See <a href="#">g.part1</a>
...	Any input arguments needed for function <a href="#">read.myacc.csv</a> if you are working with a non-standard csv formatted files. Further, any argument used in the previous version of <a href="#">g.readaccfile</a> , which will now be used to overrule the arguments specified with the parameter objects.

**Value**

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

**Author(s)**

Vincent T van Hees <[v.vanhees@accelting.com](mailto:v.vanhees@accelting.com)>

**Examples**

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

## End(Not run)
```

---

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

---

### Description

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

### Usage

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

### Arguments

<code>datafile</code>	Full path to the folder where the movisens bin files are stored. Note that movisens store a set of bin file in one folder per recording. GGIR will read the pertinent bin file to access to the temperature data.
<code>desiredtz</code>	See <a href="#">g.getmeta</a>
<code>from</code>	Origin point to derive the temperature from movisens files (automatically calculated by GGIR)
<code>to</code>	End point to derive the temperature from movisens files (automatically calculated by GGIR)
<code>interpolationType</code>	Integer, see <a href="#">g.getmeta</a>

### Value

Data matrix with the temperature values resampled at 64 Hz.

### Examples

```
## Not run:
  P = g.readtemp_movisens(datafile, desiredtz = "", from = c(), to = c())

## End(Not run)
```

---

g.report.part2      *Generate report from milestone data produced by [g.part2](#)*

---

**Description**

Creates report from milestone data produced by [g.part2](#). Not intended for direct use by package user

**Usage**

```
g.report.part2(metadataadir=c(), f0=c(), f1=c(), maxdur = 0,
selectdaysfile=c(), store.long=FALSE)
```

**Arguments**

metadataadir	see <a href="#">g.part2</a>
f0	see <a href="#">g.part2</a>
f1	see <a href="#">g.part2</a>
maxdur	see <a href="#">g.part2</a>
selectdaysfile	see <a href="#">g.part2</a>
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.

**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](mailto: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(),metadataadir=c(),loglocation = c(),f0=c(),
f1=c(),storefolderstructure=TRUE, data_cleaning_file=c(), sleepwindowType = "SPT")
```

**Arguments**

datadir	see <a href="#">g.part4</a>
metadatadir	see <a href="#">g.part4</a>
loglocation	see <a href="#">g.part4</a>
f0	see <a href="#">g.part4</a>
f1	see <a href="#">g.part4</a>
storefolderstructure	
	see <a href="#">g.part4</a>
data_cleaning_file	
	see <a href="#">g.part4</a>
sleepwindowType	
	see <a href="#">g.part4</a>

**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](mailto: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(metadataadir=c(),f0=c(),f1=c(),loglocation=c(),
               includenightcrit=c(),includedaycrit=c(),
               data_cleaning_file=c(),
               includedaycrit.part5=2/3,
               minimum_MM_length.part5=23,
               week_weekend_aggregate.part5=FALSE,
               LUX_day_segments=c())
```

### Arguments

metadataadir	see <a href="#">g.part5</a>
f0	see <a href="#">g.part5</a>
f1	see <a href="#">g.part5</a>
loglocation	see <a href="#">g.part4</a>
includenightcrit	Despricated as of version 2.0, not used anymore in part 5 report
includedaycrit	Despricated as of version 2.0, not used anymore in part 5 report
data_cleaning_file	see <a href="#">g.part4</a>
includedaycrit.part5	Inclusion criteria for number of valid hours, either as expressed as a ratio of 1 or as the number of hours in a 24 hour day.
minimum_MM_length.part5	Minimum length in hours of a MM day to be included in the cleaned part 5 results.
week_weekend_aggregate.part5	Boolean to indicate whether week and weekend-days aggregates should be stored. This is turned off by default as it generates a large number of extra columns in the output report.
LUX_day_segments	see <a href="#">g.part5</a>

**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\_dayssummary\_\* part5\_personsummary\_\*

The following files are stored in the folder results/QC: part5\_dayssummary\_full\_\*

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

**Author(s)**

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

---

g.shell.GGIR

*Shell function for analysing an accelerometer dataset.*

---

**Description**

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

**Usage**

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

**Arguments**

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

do.report	For which parts to generate a summary spreadsheet: 2 and/or 4. Default is c(2). A report will be generated based on the available milestone data. When creating milestone data with multiple machines it is advisable to turn the report generation off when generating the milestone data, value = c(), and then to merge the milestone data and turn report generation back on while setting overwrite to FALSE.
configfile	Configuration file previously generated by g.shell.GGIR. See details.
myfun	External function object to be applied to raw data, see <a href="#">g.getmeta</a> .
...	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 function documentation for <a href="#">g.part1</a> , <a href="#">g.part2</a> , and <a href="#">g.part3</a> . You cannot provide these objects as argument to g.shell.GGIR, but you can provide the parameters inside them as input to g.shell.GGIR.

### Details

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

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

Note: When combining a configuration file with explicitly provided argument values, the explicitly 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 parameter values which can be inspected with command `print(load_params())`, and if you are specifically interested in a certain subgroup of parameters, let's say 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.

### 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](mailto: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)
datadir= "C:/myfolder/mydata"
outputdir= "C:/myresults"
studyname="test"
f0 = 1
f1 = 2
g.shell.GGIR(#-----
             # General parameters
             #-----
             mode=mode,
             datadir=datadir,
             outputdir=outputdir,
             studyname=studyname,
             f0=f0,
             f1=f1,
             overwrite = FALSE,
             do.imp=TRUE,
             idloc=1,
             print.filename=FALSE,
             storefolderstructure = FALSE,
             #-----
             # Part 1 parameters:
             #-----
             window sizes = c(5,900,3600),
             do.cal=TRUE,
             do.enmo = TRUE,
             do.anglez=TRUE,
             chunksize=1,
             printsummary=TRUE,
             #-----
             # Part 2 parameters:
             #-----
             strategy = 1,
             ndayswindow=7,
             hrs.del.start = 1,
             hrs.del.end = 1,
             maxdur = 9,
             includedaycrit = 16,
             L5M5window = c(0,24),
             M5L5res = 10,
             winhr = c(5,10),
             qllevels = c(c(1380/1440),c(1410/1440)),
```



```

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

## End(Not run)

```

**Description**

Detects sustained inactivity bouts. Function not intended for direct use by package user

**Usage**

```
g.sib.det(M, IMP, I, twd = c(-12, 12),
          acc.metric = "ENMO", desiredtz = "",
          myfun=c(), sensor.location = "wrist", params_sleep = c(), ...)
```

**Arguments**

M	Object produced by <a href="#">g.getmeta</a>
IMP	Object produced by <a href="#">g.impute</a>
I	Object produced by <a href="#">g.inspectfile</a>
twd	Vector of length 2, indicating the time window to consider as hours relative to midnight.
acc.metric	Which one of the metrics do you want to consider to analyze L5. The metric of interest need to be calculated in M (see <a href="#">g.part1</a> )
desiredtz	See <a href="#">g.part3</a>
myfun	External function object to be applied to raw data. See details <a href="#">applyExtFunction</a> .
sensor.location	Character to indicate sensor location, default is wrist. If it is hip HDCZA algorithm also requires longitudinal axis of sensor to be between -45 and +45 degrees.
params_sleep	See <a href="#">g.part3</a>
...	Any argument used in the previous version of g.sib.det, which will now be used to overrule the arguments specified with the parameter objects.

**Value**

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

**Author(s)**

Vincent T van Hees <[v.vanhees@accelting.com](mailto:v.vanhees@accelting.com)>

---

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

---

### Description

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

### Usage

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

### Arguments

SLE	Output from <a href="#">g.sib.det</a>
M	Output from <a href="#">g.getmeta</a>
I	Output from <a href="#">g.inspectfile</a>
plottitle	Title to be used in the plot
nightsperpage	Number of nights to show per page
desiredtz	See <a href="#">g.part3</a>

### Value

Function has no output other than the plot

### Author(s)

Vincent T van Hees <[v.vanhees@accelting.com](mailto:v.vanhees@accelting.com)>

---

g.sib.sum                      *sustained inactivity bouts detection*

---

### Description

Detects sustained inactivity bouts. Function not intended for direct use by package user

### Usage

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

**Arguments**

SLE	Output from <a href="#">g.sib.det</a>
M	Object produced by <a href="#">g.getmeta</a>
ignorenonwear	If TRUE then ignore detected monitor non-wear periods to avoid confusion between monitor non-wear time and sustained inactivity (default = TRUE)
desiredtz	See <a href="#">g.part3</a>

**Value**

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

**Author(s)**

Vincent T van Hees <[v.vanhees@accelting.com](mailto:v.vanhees@accelting.com)>

---

g.sibreport	<i>Generate sustained inactivity bouts report</i>
-------------	---

---

**Description**

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

**Usage**

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

**Arguments**

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

**Value**

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

**Author(s)**

Vincent T van Hees <[v.vanhees@accelting.com](mailto:v.vanhees@accelting.com)>

---

g.wavread	<i>function to read .wav files as produced by the accelerometer named 'Axivity'</i>
-----------	---

---

### Description

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

### Usage

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

### Arguments

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

### Details

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

### Value

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

### Author(s)

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

---

g.weardec	<i>Detects whether accelerometer is worn</i>
-----------	--

---

### Description

Uses the object produced by [g.part1](#) to assess whether the accelerometer was worn

### Usage

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

**Arguments**

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

**Value**

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

**Author(s)**

Vincent T van Hees <[v.vanhees@accelting.com](mailto:v.vanhees@accelting.com)>

**Examples**

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

---

`getFirstTimestamp`      *Extract first timestamp from GENEActiv file*

---

**Description**

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

**Usage**

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

**Arguments**

f	GENEActiv filename
p1	First value of timestamps object

**Value**

POSIX object with `starttime`

**Author(s)**

Joe Heywood <[j.heywood@ucl.ac.uk](mailto:j.heywood@ucl.ac.uk)>

---

getfolderstructure      *Extracts folderstructure based on data directory.*

---

### Description

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

### Usage

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

### Arguments

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

### Value

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

### Examples

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

---

getStartEnd              *Generate start and end time of a day*

---

### Description

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

### Usage

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

**Arguments**

d	character with date (without time) format
startHour	Hour that analysis starts at
outputFormat	Characterstring indicating outputFormat
tz	Same as desiredtz in <a href="#">g.part1</a>

**Value**

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

**Author(s)**

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

**Examples**

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

---

getStartEndNumeric      *Generate start and end page of a day*

---

**Description**

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

**Usage**

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

**Arguments**

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

**Value**

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

**Author(s)**

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



## Examples

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

---

get\_nw\_clip\_block\_params

*Set monitor brand specific parameters*

---

## Description

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

## Usage

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

## Arguments

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

## Author(s)

Vincent T van Hees <[v.vanhees@accelting.com](mailto:v.vanhees@accelting.com)>

---

get\_starttime\_weekday\_meantemp\_truncdata

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

---

### Description

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

### Usage

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

### Arguments

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

### Author(s)

Vincent T van Hees <[v.vanhees@accelting.com](mailto:v.vanhees@accelting.com)>

---

HASIB

*Heuristic algorithms for sustained inactivity bouts detection*

---

### Description

Apply heuristic algorithms for sustained inactivity bouts detection. Function not intended for direct use by package user

### Usage

```
HASIB(HASIB.algo = "vanHees2015", timethreshold=c(), anglethreshold=c(),  
      time=c(), anglez=c(), ws3=c(), zeroCrossingCount=c(), BrondCount = c())
```

### Arguments

HASIB.algo	Character to indicator which sib algorithm should be used. Default value: "vanHees2015". Other options: "Sadeh1994", "Galland2012"
anglethreshold	See <a href="#">g.sib.det</a>
timethreshold	See <a href="#">g.sib.det</a>
time	Vector with time per short epoch
anglez	Vector with z-angle per short epoch
ws3	See <a href="#">g.getmeta</a>
zeroCrossingCount	Vector with zero crossing counts per epoch as required for Sadeh algorithm
BrondCount	Vector with Brond counts per epoch to be used by the Sadeh algorithm

### Value

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

### Author(s)

Vincent T van Hees <[v.vanhees@accelting.com](mailto:v.vanhees@accelting.com)>

---

 HASPT

*Heuristic Algorithms estimating SPT window.*


---

### Description

As used in function [g.sib.det](#). Function is not intended for direct use by GGIR user.

### Usage

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

### Arguments

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

### Value

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

### Author(s)

Vincent T van Hees <[v.vanhees@accelting.com](mailto:v.vanhees@accelting.com)>

---

identify_levels	<i>Identifies levels of behaviour for g.part5 function.</i>
-----------------	---

---

### Description

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

### Usage

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

### Arguments

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

### Value

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

### Author(s)

Vincent T van Hees <[v.vanhees@accelting.com](mailto:v.vanhees@accelting.com)>

### Examples

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

---

is.ISO8601	<i>Check whether character timestamp is in iso8601 format.</i>
------------	--

---

**Description**

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

**Usage**

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

**Arguments**

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

**Examples**

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

---

isfilelist	<i>Checks whether datadir is a directory or a vector with filenames</i>
------------	---

---

**Description**

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

**Usage**

```
isfilelist(datadir)
```

**Arguments**

datadir	Argument datadir as used in various other functions in GGIR
---------	---

**Value**

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

**Examples**

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

## End(Not run)
```

---

ismovisens	<i>Checks whether the files to process are collected with movisens accelerometers.</i>
------------	--

---

### Description

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

### Usage

```
ismovisens(data)
```

### Arguments

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

### Value

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

### Examples

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

## End(Not run)
```

---

iso8601chartime2POSIX	<i>Convert iso8601 timestamps to POSIX timestamp</i>
-----------------------	--

---

### Description

To avoid ambiguities when sharing and comparing timestamps. All timestamps are expressed in iso8601 format: [https://en.wikipedia.org/wiki/ISO\\_8601](https://en.wikipedia.org/wiki/ISO_8601) However, to generate plots in R we need to convert them back to POSIX

### Usage

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

**Arguments**

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

**Examples**

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

---

is\_this\_a\_dst\_night    *Check whether the night starting on a calendar date has DST.*

---

**Description**

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

**Usage**

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

**Arguments**

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

**Value**

dst_night_or_not	If value=0 no DST, if value=1 time moved forward, if value=-1 time moved forward
dsthour	Either the double hour or the hour that was skipped, this differs between countries

**Examples**

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



---

load_params	<i>Load default parameters</i>
-------------	--------------------------------

---

**Description**

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

**Usage**

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

**Arguments**

group            Character vector with parameter groups to be loaded.

**Value**

Lists of parameter objects

**Author(s)**

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

---

numUnpack	<i>Simple function using Rcpp</i>
-----------	-----------------------------------

---

**Description**

Simple function using Rcpp

**Usage**

```
numUnpack(pack)
```

**Arguments**

pack            vector of integer

**Examples**

```
## Not run:  
  numUnpack()  
  
## End(Not run)
```

---

 parseGT3Xggir

---

*Parse activity samples from a GT3X file with batch loading*


---

### Description

Parse activity samples from a GT3X file. The code in this function is a modified version of the `read.gt3x` in that it aids batch-loading of modern `gt3x` files. A pull request has been made to feed these enhancements back into the original code base <https://github.com/THLfi/read.gt3x/pull/40>. If and when merged we intend to deprecate the GGIR version of the code and make a direct dependency.

### Usage

```

parseGT3Xggir(
  filename,
  max_samples,
  scale_factor,
  sample_rate,
  start_time,
  batch_begin = 0L,
  batch_end = 0L,
  verbose = FALSE,
  debug = FALSE,
  impute_zeroes = FALSE
)

```

### Arguments

<code>filename</code>	(char*) path to a <code>log.bin</code> file inside the unzipped <code>gt3x</code> folder, which contains the activity samples
<code>max_samples</code>	Maximum number of rows to parse. The returned matrix will always contain this number of rows, having zeroes if not data is found.
<code>scale_factor</code>	Scale factor for the activity samples.
<code>sample_rate</code>	sampling rate for activity samples.
<code>start_time</code>	starting time of the sample recording.
<code>batch_begin</code>	first second in time relative to start of raw non-imputed recording to include in this batch
<code>batch_end</code>	last second in time relative to start of raw non-imputed recording to include in this batch
<code>verbose</code>	Print the parameters from the <code>log.bin</code> file and other messages?
<code>debug</code>	Print information for every activity second
<code>impute_zeroes</code>	Impute zeros in case there are missingness?

**Value**

Returns a matrix with `max_samples` rows and 3 columns with the acceleration samples. The matrix has attributes "time\_index", "missingness", "start\_time\_log", "sample\_rate", "impute\_zeroes".

---

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](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 <a href="https://en.wikipedia.org/wiki/List_of_tz_databases">https://en.wikipedia.org/wiki/List_of_tz_databases</a> for full list

**Author(s)**

Vincent T van Hees <[v.vanhees@accelting.com](mailto: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.gt3x\_ggir      *Read GT3X*

---

**Description**

Read activity samples from a GT3X file as a matrix. Please note that all timestamps are in local time (of the device) even though they are represented as POSIXct with GMT timezone.

The code in this function is a modified version of the `read.gt3x` in that it aids batch-loading of modern gt3x files. A pull request has been made to feed these enhancements back into the original code base <https://github.com/THLfi/read.gt3x/pull/40>. If and when merged we intend to deprecate the GGIR version of the code and make a direct dependency.

**Usage**

```
read.gt3x_ggir(
  path,
  verbose = FALSE,
  asDataFrame = FALSE,
  imputeZeroes = FALSE,
  flag_idle_sleep = FALSE,
  cleanup = FALSE,
  ...,
  add_light = FALSE
)
```

**Arguments**

path	Path to gt3x folder
verbose	print diagnostic messages
asDataFrame	convert to an activity_df, see as.data.frame.activity
imputeZeroes	Impute zeros in case there are missingness? Default is FALSE, in which case the time series will be incomplete in case there is missingness.
flag_idle_sleep	flag idle sleep mode. If imputeZeroes = TRUE, this finds where all 3 axes are zero.
cleanup	should any unzipped files be deleted?
...	additional arguments to pass to parseGT3X C++ code
add_light	add light data to the data.frame if data exists in the GT3X

**Value**

A numeric matrix with 3 columns (X, Y, Z) and the following attributes:

- `start_time` : Start time from info file in POSIXct format.
- `subject_name` : Subject name from info file
- `time_zone` : Time zone from info file
- `missingness` : Named integer vector. Names are POSIXct timestamps and values are the number of missing values.

**Note**

The timestamps in the .gt3x data format are saved in .NET format, which is nanoseconds in local time since 0001-01-01. This is a bit tricky to parse into an R datetime format. DateTimes are therefore represented as POSIXct format with the 'GMT' timezone attribute, which is false; the datetime actually represents local time.

---

read.myacc.csv	<i>Read custom csv files with accelerometer data</i>
----------------	--

---

## Description

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

## Usage

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

## Arguments

rmc.file	Filename of file to be read.
rmc.nrow	Number of rows to read, same as nrow argument in read.csv and in fread.
rmc.skip	Number of rows to skip, same as skip argument in read.csv and in fread.
rmc.dec	Decimal used for numbers, same as skip argument in read.csv and in fread.
rmc.firstrow.acc	First row (number) of the acceleration data.
rmc.firstrow.header	First row (number) of the header. Leave blank if the file does not have a header.
rmc.header.length	If file has header, specify header length (numeric).

<code>rmc.col.acc</code>	Vector with three column (numbers) in which the acceleration signals are stored
<code>rmc.col.temp</code>	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 <code>g.calibrate</code> .
<code>rmc.col.time</code>	Scalar with column (number) in which the timestamps are stored. Leave in default setting if timestamps are not stored.
<code>rmc.unit.acc</code>	Character with unit of acceleration values: "g", "mg", or "bit"
<code>rmc.unit.temp</code>	Character with unit of temperature values: (K)elvin, (C)elsius, or (F)ahrenheit
<code>rmc.unit.time</code>	Character with unit of timestamps: "POSIX", "UNIXsec" (seconds since origin, see argument <code>origin</code> ), "character", or "ActivPAL" (exotic timestamp format only used in the ActivPAL activity monitor).
<code>rmc.format.time</code>	Format of timestamp, only used for <code>rmc.unit.time</code> : character and POSIX.
<code>rmc.bitrate</code>	Numeric: If unit of acceleration is a bit then provide bit rate, e.g. 12 bit.
<code>rmc.dynamic_range</code>	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.
<code>rmc.unsignedbit</code>	Boolean, if <code>unsignedbit = TRUE</code> means that bits are only positive numbers. if <code>unsignedbit = FALSE</code> then bits are both positive and negative.
<code>rmc.origin</code>	Origin of time when unit of time is UNIXsec, e.g. 1970-1-1
<code>rmc.desiredtz</code>	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 <code>configtz</code> to specify where the device was configured (not implemented yet).
<code>rmc.sf</code>	Sample rate in Hertz, if this is stored in the file header then the that will be used instead.
<code>rmc.headername.sf</code>	If file has a header: Row name (character) under which the sample frequency can be found.
<code>rmc.headername.sn</code>	If file has a header: Row name (character) under which the serial number can be found.
<code>rmc.headername.recordingid</code>	If file has a header: Row name (character) under which the recording ID can be found.
<code>rmc.header.structure</code>	Character used to split the header name from the header value, e.g. ":" or " "
<code>rmc.check4timegaps</code>	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 <code>TRUE</code> the the gaps in time will be filled with zeros.

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

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

`interpolationType`  
See [g.getmeta](#)

### 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 [g.shell.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.

### Value

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

### Author(s)

Vincent T van Hees <[v.vanhees@accelting.com](mailto:v.vanhees@accelting.com)>

### Examples

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

---

resample

*Simple function using Rcpp*

---

### Description

Simple function using Rcpp

### Usage

```
resample(raw, rawTime, time, stop, type=1)
```

**Arguments**

raw	stop-by-3 matrix with raw values of x, y and z.
rawTime	vector with stop elements of raw time.
time	array with required time points.
stop	Number of rows in raw
type	integer to indicate type of interpolation, 1=linear, 2=nearest neighbour

**Examples**

```
## Not run:
  resample()

## End(Not run)
```

---

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

---

**Description**

Function queries available memory to either lower or increase the blocksize used by function [g.readaccfile](#)

**Usage**

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

**Arguments**

blocksize	Number of filepages (binary data) or rows (other dataformats).
bsc_qc	Data.frame with columns time (timestamp from Sys.time) and size (memory size). This is used for housekeeping in <a href="#">g.calibrate</a> and <a href="#">g.getmeta</a>

**Value**

List with blocksize and bsc\_qc, same format as input, although bsc\_qc has one new row.



# Index

- \* **datasets**
  - data.calibrate, [12](#)
  - data.getmeta, [12](#)
  - data.inspectfile, [13](#)
- applyExtFunction, [6](#), [9](#), [38](#), [46](#), [52](#), [56](#), [82](#)
- CalcSleepRegularityIndex, [7](#)
- chartime2iso8601, [8](#)
- check\_myfun, [8](#)
- check\_params, [9](#)
- create\_test\_acc\_csv, [10](#)
- create\_test\_sleeplog\_csv, [11](#)
- createConfigFile, [10](#)
- data.calibrate, [12](#)
- data.getmeta, [12](#)
- data.inspectfile, [13](#)
- datadir2fnames, [13](#)
- extract\_params, [9](#), [14](#)
- g.abr.day.names, [15](#)
- g.analyse, [15](#), [17–21](#), [29](#), [34](#), [36](#), [51](#)
- g.analyse.avday, [20](#), [21](#)
- g.analyse.avday(g.analyse.avy), [17](#)
- g.analyse.avy, [17](#)
- g.analyse.perday, [18](#), [20](#), [21](#)
- g.analyse.perfile, [20](#)
- g.applymetrics, [22](#), [46](#), [47](#)
- g.binread, [5](#), [23](#)
- g.calibrate, [5](#), [12](#), [16](#), [20](#), [24](#), [45](#), [48](#), [72](#), [89](#), [104](#)
- g.conv.actlog, [25](#), [54](#)
- g.convert.part2.long, [26](#)
- g.create.sp.mat, [27](#)
- g.createcoordinates, [27](#)
- g.cwaread, [5](#), [28](#)
- g.detectmidnight, [17–19](#), [29](#)
- g.dotorcomma, [30](#), [35](#), [39](#), [73](#), [89](#), [90](#)
- g.downsample, [31](#)
- g.extractheadervars, [19](#), [20](#), [31](#)
- g.fragmentation, [32](#), [53](#)
- g.getbout, [34](#), [53](#), [54](#)
- g.getidfromheaderobject, [35](#)
- g.getM5L5, [36](#), [55](#)
- g.getmeta, [6](#), [7](#), [9](#), [12](#), [15–17](#), [19](#), [21](#), [28](#), [30](#), [37](#), [39](#), [40](#), [42](#), [45](#), [64](#), [69](#), [70](#), [72–74](#), [79](#), [82–84](#), [86](#), [89–91](#), [103](#), [104](#)
- g.getstarttime, [39](#)
- g.impute, [15–17](#), [19](#), [20](#), [39](#), [49](#), [51](#), [70](#), [82](#)
- g.imputeTimegaps, [41](#)
- g.inspectfile, [13](#), [16](#), [20](#), [32](#), [35](#), [39](#), [40](#), [41](#), [70](#), [73](#), [82](#), [83](#), [89](#), [90](#)
- g.intensitygradient, [42](#)
- g.IVIS, [21](#), [43](#), [55](#)
- g.loadlog, [44](#), [57](#), [84](#)
- g.part1, [6](#), [13](#), [16](#), [19](#), [24](#), [37–40](#), [42](#), [45](#), [45](#), [51](#), [52](#), [55–57](#), [59](#), [61](#), [72](#), [73](#), [78](#), [79](#), [82](#), [85](#), [87–89](#), [103](#)
- g.part2, [16](#), [17](#), [19](#), [29](#), [45](#), [51](#), [57](#), [59](#), [61](#), [72](#), [73](#), [75](#), [78](#), [79](#), [93](#)
- g.part3, [56](#), [59](#), [61](#), [64](#), [78](#), [79](#), [82–84](#)
- g.part4, [44](#), [59](#), [60](#), [66](#), [75–78](#)
- g.part4\_extractid, [60](#)
- g.part5, [32](#), [61](#), [62](#), [63](#), [65](#), [66](#), [68](#), [69](#), [72](#), [77](#), [84](#)
- g.part5.addfirstwake, [62](#)
- g.part5.addsib, [63](#)
- g.part5.classifyNaps, [57](#), [58](#), [64](#)
- g.part5.definedays, [65](#)
- g.part5.fixmissingnight, [66](#)
- g.part5.handle\_lux\_extremes, [66](#)
- g.part5.lux\_persegment, [67](#)
- g.part5.onsetwaketiming, [68](#)
- g.part5.savetimeseries, [68](#)
- g.part5.wakesleepwindows, [69](#)
- g.plot, [27](#), [70](#)
- g.plot5, [15](#), [53](#), [71](#)
- g.readaccfile, [30](#), [39](#), [72](#), [90](#), [104](#)

`g.readtemp_movisens`, 74  
`g.report.part2`, 75  
`g.report.part4`, 75  
`g.report.part5`, 50, 53, 69, 77  
`g.shell.GGIR`, 6, 10, 45, 46, 51, 52, 56, 71, 72, 78, 89, 103  
`g.sib.det`, 7, 50, 81, 83, 84, 91, 92  
`g.sib.plot`, 83  
`g.sib.sum`, 57, 83  
`g.sibreport`, 64, 84  
`g.wavread`, 28, 85  
`g.weardec`, 17, 19, 85  
`get_nw_clip_block_params`, 46, 49, 89  
`get_starttime_weekday_meantemp_truncdata`, 90  
`getFirstTimestamp`, 86  
`getfolderstructure`, 87  
`getStartEnd`, 87  
`getStartEndNumeric`, 88  
GGIR (GGIR-package), 4  
GGIR-package, 4  
  
HASIB, 57, 91  
HASPT, 57, 92  
  
`identify_levels`, 32, 93  
`is.ISO8601`, 94  
`is_this_a_dst_night`, 96  
`isfilelist`, 13, 94  
`ismovisens`, 95  
`iso8601chartime2POSIX`, 95  
  
`load_params`, 97  
  
`numUnpack`, 97  
  
`parseGT3Xggir`, 98  
`POSIXtime2iso8601`, 99  
  
`read.gt3x_ggir`, 99  
`read.myacc.csv`, 30, 46, 49, 73, 89, 101  
`resample`, 103  
  
`updateBlocksize`, 104