Package ‘GGIRread’

June 3, 2024

Type Package

Title Wearable Accelerometer Data File Readers

Version 1.0.1

Date 2024-06-03

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

Description Reads data collected from wearable acceleratometers as used in sleep and physical activity research. Currently supports file formats: binary data from 'GENEActiv' <https://activinsights.com/>, .bin-format from GENEA devices (not for sale), and .cwa-format from 'Axivity' <https://axivity.com>. Primarily designed to complement R package GGIR <https://CRAN.R-project.org/package=GGIR>.

URL https://github.com/wadpac/GGIRread/

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

License Apache License (== 2.0)

Suggests testthat

Imports matlab, bitops, Rcpp (>= 0.12.10)

Depends stats, utils, R (>= 3.5.0)

NeedsCompilation yes

LinkingTo Rcpp

ByteCompile yes

Author Vincent T van Hees [aut, cre],
Patrick Bos [aut] (<https://orcid.org/0000-0002-6033-960X>),
Lena Kushleyeva [ctb],
Jing Hua Zhao [ctb],
Evgeny Mirkes [ctb],
Dan Jackson [ctb],
Jairo H Migueles [ctb],
Medical Research Council UK [cph, fnd],
Accelting [cph, fnd]

Repository CRAN

Date/Publication 2024-06-03 07:10:02 UTC

R topics documented:

`GGIRread-package` *A package to read data files from wearable accelerometers*

Description

This package offers a collection of functions to read data files from wearable accelerometers. Some functions were migrated from R package GGIR to make GGIR more modular and to reduce it's complexity, while other functions such as readGENEActiv (R) and GENEActivReader (C++) are newly written.

Details

Package:	GGIRread
Type:	Package
Version:	1.0.0
Date:	2024-03-27
License:	LGPL (>= 2.0, < 3)

Author(s)

- Vincent T van Hees <v.vanhees@accelting.com> main creator and developer
- Patrick Bos developed function GENEActivReader
- Jing Hua Zhao <jinghua.zhao@mrc-epid.cam.ac.uk> co-developed function readGenea
- Evgeny Mirkes created function readAxivity
- Dan Jackson helped improve function readAxivity

`GENEActivReader` *Function (C++) to read binary files as produced by the GENEActiv accelerometer*

Description

Function to read binary files as produced by the GENEActiv accelerometer by ActivInsights Ltd https://activinsights.com/

Usage

```
GENEActivReader(filename, start = 0L, end = 0L, progress_bar = FALSE)
```

Arguments

`filename`	filename (required)
`start`	start page for reading data
`end`	end page for reading data
`progress_bar`	Boolean

Details

If only `start` page is defined then all data is read beyond `start` until the end of the file is reached

Value

`info`	List with ReadOK (good=0 or error=1), ReadErrors (Count of pages with read errors), SampleRate (Hertz), numBlocksTotal
`time`	Numeric vector with time in miliseconds since start page
`x`	Numeric vector with x-axis acceleration in gravitational units
`y`	Numeric vector with y-axis acceleration in gravitational units
`z`	Numeric vector with z-axis acceleration in gravitational units
`temperature`	matrix with battery voltage and corresponding timestamps
`lux`	Numeric vector with lux values in Volts

Author(s)

Patrick Bos <egpbos@gmail.com> Vincent T van Hees <v.vanhees@accelting.com>

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

Description

For reading .cwa-format data with the Axivity AX3 and AX6 sensors.

Usage

```
readAxivity(filename, start = 0, end = 0, progressBar = FALSE,
  desiredtz = "", configtz = c(), interpolationType=1, loadbattery = FALSE,
  header = NULL, frequency_tol = 0.1, maxAllowedCorruptBlocks = 20)
```

Arguments

`filename` filename (required) is name of cwa file to read

`start` non-negative integer which is a cwa file block number.

`end` non-negative integer which is a cwa file block number. End must be not be less than start. If end is less or equal to start, then there is no data read.

`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, a character with timezone database name.

`configtz` 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` Integer to indicate type of interpolation to be used, 1=linear, 2=nearest neighbour.

`loadbattery` Boolean to indicate whether battery voltage should be loaded

`header` Header to be reused if it was extracted earlier

`frequency_tol` Numeric value representing the tolerance for frequency bias expressed as a fraction of the expected sampling frequency. For example, 0.10 indicates that we accept the sampling frequency configured at 100 Hertz to vary between 90 and 110 Hertz. If this condition is not met the data will be imputed with a constant value and the start and end time will be logged in the QClog output object. Regardless of the setting the function will always log instances when frequency differs by more than 5%, but if this is less than frequency_tol the block will not be imputed.

`maxAllowedCorruptBlocks` Max number of consecutive blocks with a failed checksum that we'll tolerate.

Value

- `data` dataframe with data from start inclusive till end exclusive. If start == end then data section of final structure is empty.
  - timestamp in numeric format
  - gx gyroscope x-axis (only available in AX6)
  - gy gyroscope y-axis (only available in AX6)
  - gz gyroscope z-axis (only available in AX6)
  - x accelerometer x-axis
  - y accelerometer y-axis
  - z accelerometer z-axis
  - temperature in Celsius (only recorded once per block)
  - battery one value per block (only recorded once per block)
  - light in LUX on (only recorded once per block)
- `header` file header. list of header information:
  - uniqueSerialCode is unque serial code of used device
  - frequency is measurement frequency. All data will be resampled for this frequency
  - start is timestamp in numeric form. To get text representation it is enough to use as.POSIXct(start, origin = "1970-01-01", tz=desiredtz)
  - device is "Axivity"
  - firmwareVersion
  - blocks is number of datablocks with 80 or 120 raw observations in each.
- `QClog` dataframe with log of integrity checks on each block including:
  - checksum_pass Boolean to indicate whether checksum was zero.
  - blockID_previous ID for previous block
  - blockID_current ID for current block
  - start_previous numeric start time of previous block
  - start_current numeric start time of current block
  - blockLengthSeconds length of previous block
  - frequency_blockheader frequency derived from block header
  - frequency_observed frequency as observed
  - imputed Boolean to indicate whether this block was inputed

Author(s)

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

Examples

```
 cwafile  = system.file("testfiles/ax3_testfile.cwa", package = "GGIRread")[1]
AX3 = readAxivity(filename = cwafile, desiredtz = "Europe/Berlin", start = 1, end = 2)
```

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

Description

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

Usage

```
readGenea(filename, start = 0, end = 0)
```

Arguments

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

Details

If only `start` is defined then `readGenea` 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>

Examples

```
geneafile  = system.file("testfiles/genea_testfile.bin", package = "GGIRread")[1]
GENEA = readGenea(filename = geneafile, start = 0, end = 2)
```

`readGENEActiv` *Function (R) to read binary files as produced by the GENEActiv accelerometer*

Description

R function wrapper around GENEActivReader to read binary files as produced by the GENEActiv accelerometer by ActivInsights Ltd https://activinsights.com/

Usage

```
readGENEActiv(filename, start = 0L, end = 0L, progress_bar = FALSE,
              desiredtz = "", configtz = NULL)
```

Arguments

`filename`	filename (required)
`start`	start page for reading data
`end`	end page for reading data
`progress_bar`	Boolean
`desiredtz`	Character, timezone database name of the timezone where the accelerometer was worn.
`configtz`	Character, timezone database name of the timezone where the accelerometer was configured. Leave NULL if equal to timezone where experiment took place.

Details

If only `start` page is defined then all data is read beyond `start` until the end of the file is reached

Value

header:

`serial_number`	Device serial number as extracted from file header
`firmware`	Firmware version
`tzone`	Time zone as extracted from file header
`ReadOK`	see GENEActivReader
`SampleRate`	matrix with battery voltage and corresponding timestamps
`ReadErrors`	matrix with battery voltage and corresponding timestamps
`numBlocksTotal`	matrix with battery voltage and corresponding timestamps
`StartTime`	Start time as extracted from file header

data.out:

`time`	Unix time in seconds with decimal places for miliseconds
`x`	see GENEActivReader
`y`	see GENEActivReader
`z`	see GENEActivReader
`light`	Ligth values express in lux units
`temperature`	Temperature in Celsius

Author(s)

Patrick Bos <egpbos@gmail.com> Vincent T van Hees <v.vanhees@accelting.com>

Examples

```
 binfile  = system.file("testfiles/GENEActiv_testfile.bin", package = "GGIRread")[1]
rdata = readGENEActiv(filename = binfile, start = 1, end = 1, desiredtz = "Europe/London")
```

`readWav` *Deprecated function to read .wav files as can be stored with OMGUI software by Axivity Ltd*

Description

Originally designed to read the wav accelerometer data format as stored with the OMGUI software by Axivity Ltd and documented here.

This function has been deprecated as wav files are hardly used, their formatting is not consistent causing ongoing challenges with keeping the keep functional, wav format not well facilitated for AX6 recording.

Usage

```
readWav(filename, start = 1, end = 100,units="minutes")
```

Arguments

`filename`	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>

`resample` *Resample timeseries*

Description

Resample single- or multi-variate time series using linear or nearest neighbour interpolation

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

```
raw = cbind(1:10, 1:10, 1:10)
rawTime = seq(0.1, 1, by = 0.1)
time = seq(0.15001, 1.05001, by = 0.1)
stop = 10
dat_lin = resample(raw, rawTime, time, stop, type = 1)
```

Index