

# Package ‘AGread’

February 17, 2019

**Title** Read Data Files from ActiGraph Monitors

**Version** 0.2.0

**Description** Standardize the process of bringing various modes of output files into R. For more information, see:

<<https://actigraph.desk.com/customer/en/portal/articles/2515800-what-do-the-different-mode-numbers-mean-in-a-csv-or-dat-file->>.

Additionally, processes are provided to read and minimally pre-process raw data from primary accelerometer and inertial measurement unit files, as well as binary .gt3x files. ActiGraph monitors are used to estimate physical activity outcomes via body-worn sensors that measure (e.g.) acceleration or rotational velocity.

**Depends** R (>= 3.1.0)

**License** GPL-3 | file LICENSE

**Imports** anytime (>= 0.3.0), binaryLogic (>= 0.3.9), data.table (>= 1.10.4), DescTools (>= 0.99.20), dplyr (>= 0.5.0), GGIR (>= 1.5.0), lubridate, magrittr (>= 1.5), reshape2 (>= 1.4.0), rlang (>= 0.2.0), seewave (>= 2.0.5), stats, stringr (>= 1.3.0), utils

**Encoding** UTF-8

**LazyData** true

**RoxygenNote** 6.1.1

**URL** <https://github.com/paulhibbing/AGread>

**BugReports** <https://github.com/paulhibbing/AGread/issues>

**Suggests** testthat

**NeedsCompilation** no

**Author** Paul R. Hibbing [aut, cre],  
Vincent T. van Hees [ctb],  
Samuel R. LaMunion [ctb]

**Maintainer** Paul R. Hibbing <paulhibbing@gmail.com>

**Repository** CRAN

**Date/Publication** 2019-02-17 07:20:03 UTC

## R topics documented:

AGread	2
AG_collapse	3
AG_meta	4
check_columns	5
check_second	5
classify_magnetometer	6
collapse_gt3x	7
get_day_of_year	8
get_duration	8
get_imu_file_meta	9
get_minute	9
get_raw_file_meta	10
get_VM	11
imu_collapse	11
imu_filter_gyroscope	12
imu_to_check	13
imu_to_collapse	13
raw_to_collapse	14
read_AG_counts	15
read_AG_IMU	16
read_AG_raw	17
read_gt3x	18
reintegrate	19
<b>Index</b>	<b>21</b>

---

AGread

*Read Data Files from ActiGraph Monitors*

---

### Description

This provides support for reading ActiGraph files of various modes into R. For more information see: <https://actigraph.desk.com/customer/en/portal/articles/2515800-what-do-the-different-mode-numbers-mean>. Functions are provided to read and minimally pre-process raw data from primary accelerometer and inertial measurement unit files. Reading binary .gt3x files is now supported as well. See <https://github.com/actigraph/GT3X-File-Format> for more information.

### Core functions

[read\\_AG\\_counts](#)  
[read\\_AG\\_raw](#)  
[read\\_AG\\_IMU](#)  
[read\\_gt3x](#)

**Examples**

```
read_AG_counts(  
  system.file(  
    "extdata",  
    "example1sec.csv",  
    package = "AGread"  
  ),  
  skip = 11  
)  
read_AG_raw(  
  system.file(  
    "extdata",  
    "TestID_LeftWrist_RAW.csv",  
    package = "AGread"  
  )  
)  
read_AG_IMU(  
  system.file(  
    "extdata",  
    "example-IMU.csv",  
    package = "AGread"  
  )  
)  
file_3x <- system.file(  
  "extdata", "example.gt3x", package = "AGread"  
)  
read_gt3x(file_3x)
```

---

AG\_collapse

*Collapse primary accelerometer data*

---

**Description**

Collapse primary accelerometer data

**Usage**

```
AG_collapse(AG, output_window_secs = 1, samp_freq, method = "default",  
  ENMO2 = NULL)
```

**Arguments**

AG                    a dataframe of raw primary accelerometer data  
output\_window\_secs    the desired epoch length; defaults to one second  
samp\_freq            The sampling frequency

method	character scalar giving the method to use for calculating ENMO, either "default" or "block"
ENMO2	vector of leftover raw values from the previous block (if applicable)

### Examples

```
data(raw_to_collapse)
AG_collapse(raw_to_collapse, 1, 80)
```

---

AG_meta	<i>Extract meta-data from file header</i>
---------	---

---

### Description

Extract meta-data from file header

### Usage

```
AG_meta(file, verbose = FALSE, ...)
```

### Arguments

file	A character scalar giving path to an automatically-generated csv file with count values
verbose	A logical scalar: Print processing updates?
...	Further arguments passed to read.csv and fread

### Examples

```
counts_file <- system.file(
  "extdata", "example1sec.csv", package = "AGread"
)
AGread::AG_meta(counts_file)
```

---

check_columns	<i>Check if the primary accelerometer file is formatted correctly</i>
---------------	---

---

**Description**

check\_columns returns a logical scalar indicating whether there is a formatting issue with the file passed as the argument. A value of TRUE indicates the test has passed, whereas FALSE indicates an issue.

**Usage**

```
check_columns(file, skip, ...)
```

**Arguments**

file	A character scalar giving path to primary accelerometer file
skip	Header length: Number of rows to skip when reading the file
...	Arguments passed to <a href="#">read.csv</a>

**Examples**

```
raw_file <-  
  system.file("extdata",  
    "TestID_LeftWrist_RAW.csv",  
    package = "AGread")  
  
check_columns(raw_file, skip = 10)
```

---

check_second	<i>Check if the IMU data start on an exact second</i>
--------------	---

---

**Description**

Check if the IMU data start on an exact second

**Usage**

```
check_second(AG)
```

**Arguments**

AG	a dataframe of IMU data
----	-------------------------

**Examples**

```
data(imu_to_check)
check_second(imu_to_check)
```

---

classify\_magnetometer *Convert magnetometer signal to cardinal direction*

---

**Description**

Convert magnetometer signal to cardinal direction

**Usage**

```
classify_magnetometer(x = "Magnetometer X", y = "Magnetometer Y",
  z = "Magnetometer Z", orientation = "vertical")
```

**Arguments**

x	x-axis magnetometer data
y	y-axis magnetometer data
z	z-axis magnetometer data
orientation	the conversion scheme to use, from c("vertical", "horizontal")

**Value**

A vector of cardinal directions assigned from the set N, NNE, NE, ENE, E, ESE, SE, SSE, S, SSW, SW, WSW, W, WNW, NW, NNW, where N, E, S, and W are north, east, south, and west, respectively.

**See Also**

[http://s3.amazonaws.com/actigraphcorp.com/wp-content/uploads/2017/11/26205750/ActiGraph\\_IMU\\_White\\_Paper.pdf](http://s3.amazonaws.com/actigraphcorp.com/wp-content/uploads/2017/11/26205750/ActiGraph_IMU_White_Paper.pdf)

**Examples**

```
data(imu_to_collapse)

X <- mean(imu_to_collapse$Magnetometer.X)
Y <- mean(imu_to_collapse$Magnetometer.Y)
Z <- mean(imu_to_collapse$Magnetometer.Z)

classify_magnetometer(X, Y, Z)
```

---

collapse_gt3x	<i>Collapse data that were read using <a href="#">read_gt3x</a></i>
---------------	---

---

## Description

Collapse data that were read using [read\\_gt3x](#)

## Usage

```
collapse_gt3x(AG, filename = "gt3x file", output_window_secs = 1,
  filter = TRUE, filter_hz = 35, verbose = FALSE, ...)
```

```
## S3 method for class 'RAW'
collapse_gt3x(AG, filename = "gt3x file",
  output_window_secs = 1, filter = TRUE, filter_hz = 35,
  verbose = FALSE, ...)
```

```
## S3 method for class 'IMU'
collapse_gt3x(AG, filename = "gt3x file",
  output_window_secs = 1, filter = TRUE, filter_hz = 35,
  verbose = FALSE, ...)
```

## Arguments

AG	The object to collapse, inheriting from class "RAW" or "IMU"
filename	character. Filename to associate with the data.
output_window_secs	the desired epoch length; defaults to one second
filter	a logical scalar: Apply a low-pass filter to gyroscope data?
filter_hz	The cutoff for the low-pass filter
verbose	A logical scalar: Print processing updates?
...	Additional arguments. Currently unused.

## Value

A data frame of collapsed data

## Examples

```
file <- system.file(
  "extdata",
  "example.gt3x",
  package = "AGread"
)
data <- read_gt3x(file)
```

```
collapse_gt3x(data$RAW)
collapse_gt3x(data$IMU)
```

---

`get_day_of_year`      *Julian Date*

---

### **Description**

A wrapper to retrieve the Julian date.

### **Usage**

```
get_day_of_year(timestamp, format = "%Y-%m-%d %H:%M:%S")
```

### **Arguments**

`timestamp`      A character vector containing timestamp information  
`format`          The date-time format of the `timestamp` vector

### **Value**

A numeric vector of Julian dates.

### **Examples**

```
key_dates <- c("2018-01-01", "2018-12-31")
get_day_of_year(key_dates, "%Y-%m-%d")
```

---

`get_duration`      *Provide the run time of processing*

---

### **Description**

Provide the run time of processing

### **Usage**

```
get_duration(timer)
```

### **Arguments**

`timer`          The initial time



**Examples**

```
timer <- proc.time()
Sys.sleep(2.2)
AGread::message_update(16, dur = get_duration(timer))
```

---

get_imu_file_meta	<i>Get file metadata (sampling frequency, start time, and samples per epoch) for IMU</i>
-------------------	--

---

**Description**

Get file metadata (sampling frequency, start time, and samples per epoch) for IMU

**Usage**

```
get_imu_file_meta(file, output_window_secs = 1)
```

**Arguments**

file	character scalar giving path to IMU file
output_window_secs	the desired epoch length, over which to average IMU data

**Examples**

```
imu_file <-
  system.file("extdata",
    "example-IMU.csv",
    package = "AGread")

get_imu_file_meta(imu_file)
```

---

get_minute	<i>Numerical Minute of the Day.</i>
------------	-------------------------------------

---

**Description**

Converts a timestamp to a numerical value between 0 (midnight) and 1439 (23:59). Seconds can be represented using a rational decimal.

**Usage**

```
get_minute(timestamp, format = "%Y-%m-%d %H:%M:%S", rational = FALSE)
```

**Arguments**

timestamp	A character vector containing timestamp information
format	The date-time format of the timestamp vector
rational	A logical scalar. Use rational number to represent seconds?

**Examples**

```
key_times <-
  paste("2018-03-15",
        c("00:00:00",
          "01:00:00",
          "12:00:00",
          "23:59:59"))

get_minute(key_times)
get_minute(key_times, rational = TRUE)
```

---

get_raw_file_meta	<i>Get file metadata (sampling frequency and timestamps) for primary accelerometer</i>
-------------------	--

---

**Description**

Get file metadata (sampling frequency and timestamps) for primary accelerometer

**Usage**

```
get_raw_file_meta(file)
```

**Arguments**

file	character scalar giving path to primary accelerometer file
------	--

**Examples**

```
raw_file <-
  system.file("extdata",
             "TestID_LeftWrist_RAW.csv",
             package = "AGread")

get_raw_file_meta(raw_file)
```

---

get_VM	<i>Calculate vector magnitude</i>
--------	-----------------------------------

---

**Description**

Calculate vector magnitude

**Usage**

```
get_VM(triaxial, verbose = FALSE)
```

**Arguments**

triaxial	a dataframe of triaxial data on which to calculate vector magnitude
verbose	print information about variable search criteria?

**Value**

a vector of vector magnitude values

**Examples**

```
data(imu_to_collapse)

vm_columns <-
  grepl("accelerometer",
        names(imu_to_collapse),
        ignore.case = TRUE)

get_VM(data.frame(imu_to_collapse)[, vm_columns])
```

---

imu_collapse	<i>Collapse raw IMU data to a specified epoch</i>
--------------	---

---

**Description**

Collapse raw IMU data to a specified epoch

**Usage**

```
imu_collapse(AG, block_size, verbose = FALSE)
```

**Arguments**

AG	dataframe containing raw IMU data
block_size	number of samples per epoch
verbose	A logical scalar: Print processing updates?

**Value**

dataframe of IMU data averaged over the specified epoch length

**Examples**

```
data(imu_to_collapse)
imu_collapse(imu_to_collapse, 100)
```

---

imu\_filter\_gyroscope *Low-Pass filter Gyroscope data*

---

**Description**

Low-Pass filter Gyroscope data

**Usage**

```
imu_filter_gyroscope(AG, samp_rate, filter_hz = 35, verbose = FALSE)
```

**Arguments**

AG	a dataframe of IMU data
samp_rate	The sampling rate, in Hz
filter_hz	The cutoff for the low-pass filter
verbose	A logical scalar: Print processing updates?

**Examples**

```
data(imu_to_collapse)
imu_filter_gyroscope(imu_to_collapse, 100)
```

---

imu_to_check	<i>IMU data to check</i>
--------------	--------------------------

---

### Description

A dataset for demonstrating checks that are applied to IMU data.

### Usage

imu\_to\_check

### Format

A data frame with 300 rows and 8 variables:

**file\_source\_IMU** The filename of the IMU file

**date\_processed\_IMU** The date the IMU file was processed

**Timestamp** The corresponding time for each row of data

**Gyroscope\_VM\_DegPerS** Gyroscope vector magnitude, in degrees per second

**mean\_abs\_Gyroscope\_x\_DegPerS** Rotation in x axis, degrees per second

**mean\_abs\_Gyroscope\_y\_DegPerS** Rotation in y axis, degrees per second

**mean\_abs\_Gyroscope\_z\_DegPerS** Rotation in z axis, degrees per second

**mean\_magnetometer\_direction** Cardinal direction of magnetometer signal, averaged over one second

---

imu_to_collapse	<i>IMU data to collapse</i>
-----------------	-----------------------------

---

### Description

A partially-processed IMU dataset ready to be collapsed from raw samples to one-second summaries.

### Usage

imu\_to\_collapse

**Format**

A data frame with 1500 rows and 17 variables:

**Timestamp** The corresponding time for each row of data

**Accelerometer.X** Secondary accelerometer x-axis data, in G

**Accelerometer.Y** Secondary accelerometer y-axis data, in G

**Accelerometer.Z** Secondary accelerometer z-axis data, in G

**Temperature** Temperature of the IMU, in Celsius

**Gyroscope.X** Gyroscope x-axis data, in degrees per second

**Gyroscope.Y** Gyroscope y-axis data, in degrees per second

**Gyroscope.Z** Gyroscope z-axis data, in degrees per second

**Magnetometer.X** Magnetometer x-axis data, in micro-Teslas

**Magnetometer.Y** Magnetometer y-axis data, in micro-Teslas

**Magnetometer.Z** Magnetometer z-axis data, in micro-Teslas

**file\_source\_IMU** The filename of the IMU file

**date\_processed\_IMU** The date the IMU file was processed

**ms** The millisecond value of the timestamp

**mean\_Accel\_VM** Vector magnitude of the secondary accelerometer signal, in G

**Gyroscope\_VM\_DegPerS** Gyroscope vector magnitude, in degrees per second

**Magnetometer\_VM\_MicroT** Vector magnitude of the magnetometer signal, in micro-Teslas

---

raw\_to\_collapse

*Primary accelerometer data to collapse*

---

**Description**

A partially-processed primary accelerometer dataset ready to be collapsed from raw samples to one-second summaries.

**Usage**

raw\_to\_collapse

**Format**

A data frame with 24000 rows and 3 variables:

**Accelerometer X** Primary accelerometer x-axis data, in G

**Accelerometer Y** Primary accelerometer y-axis data, in G

**Accelerometer Z** Primary accelerometer z-axis data, in G

---

read_AG_counts	<i>Read data table files containing count values</i>
----------------	--

---

**Description**

Read data table files containing count values

**Usage**

```
read_AG_counts(file, verbose = FALSE, skip = 10, nrows = 10,  
              header = FALSE, ...)
```

**Arguments**

file	A character scalar giving path to an automatically-generated csv file with count values
verbose	A logical scalar: Print processing updates?
skip	Header length: Number of rows to skip when reading the file
nrows	Header length: Number of rows to read when retrieving meta-data
header	A logical scalar: Are variable names contained in first row of file?
...	Further arguments passed to read.csv and fread

**Value**

A data frame reflecting the data contained in the csv file

**Examples**

```
read_AG_counts(  
  system.file(  
    "extdata",  
    "example1sec.csv",  
    package = "AGread"  
  ),  
  skip = 11  
)
```

---

read\_AG\_IMU                      *File reading function for IMU files*

---

### Description

File reading function for IMU files

### Usage

```
read_AG_IMU(file, output_window_secs = 1, verbose = FALSE, skip = 10,  
            filter = TRUE, filter_hz = 35, output_vars = c("accelerometer",  
                  "temperature", "gyroscope", "magnetometer"))
```

### Arguments

file	character scalar giving the path to the IMU file
output_window_secs	the desired epoch length; defaults to one second
verbose	A logical scalar: Print processing updates?
skip	Header length: Number of rows to skip when reading the file
filter	a logical scalar: Apply a low-pass filter to gyroscope data?
filter_hz	The cutoff for the low-pass filter
output_vars	character. IMU variables to include in output.

### Value

A dataframe giving processed IMU data in the specified epoch length

### Examples

```
imu_file <- system.file(  
  "extdata",  
  "example-IMU.csv",  
  package = "AGread"  
)  
  
read_AG_IMU(imu_file)
```



---

read_AG_raw	<i>File reading function for primary accelerometer files</i>
-------------	--

---

### Description

File reading function for primary accelerometer files

### Usage

```
read_AG_raw(file, output_window_secs = 1, calibrate = FALSE,  
            verbose = FALSE, skip = 10, block = FALSE, return_raw = FALSE, ...)
```

### Arguments

file	A character scalar giving path to primary accelerometer file
output_window_secs	the desired epoch length; defaults to one second
calibrate	logical. Perform autocalibration using <a href="#">g.calibrate</a>
verbose	A logical scalar: Print processing updates?
skip	Header length: Number of rows to skip when reading the file
block	logical. Should file be read in blocks? Will be automatically invoked if file is larger than 2 GB.
return_raw	logical. Return raw triaxial data?
...	Arguments passed to read.csv in <a href="#">check_columns</a>

### Value

A dataframe giving processed raw data from the primary accelerometer in the specified epoch length

### Examples

```
raw_file <- system.file(  
  "extdata",  
  "TestID_LeftWrist_RAW.csv",  
  package = "AGread"  
)  
  
read_AG_raw(raw_file)
```

---

read_gt3x	<i>Read data from a gt3x file</i>
-----------	-----------------------------------

---

### Description

Read data from a gt3x file

### Usage

```
read_gt3x(file, tz = "UTC", verbose = FALSE, give_timestamp = TRUE,
  include = c("METADATA", "PARAMETERS", "SENSOR_SCHEMA", "BATTERY", "EVENT",
    "TAG", "ACTIVITY", "HEART_RATE_BPM", "HEART_RATE_ANT", "HEART_RATE_BLE",
    "LUX", "CAPSENSE", "EPOCH", "EPOCH2", "EPOCH3", "EPOCH4", "ACTIVITY2",
    "SENSOR_DATA"))
```

### Arguments

file	character. Path to the file
tz	character. The timezone to use
verbose	logical. Print updates to console?
give_timestamp	logical. Include timestamp in output?
include	character. The PACKET types to parse

### Details

The default value for include gives all possible packet types, of which there are 18. Processing time can be reduced by passing a subset of the 18 possibilities. Exclusion is not recommended for the PARAMETERS and SENSOR\_SCHEMA packets, which also do not take long to process.

### Value

A list of processed data, with one element for each of the relevant packet types.

### References

<https://github.com/actigraph/GT3X-File-Format>

### Examples

```
file_3x <- system.file(
  "extdata", "example.gt3x", package = "AGread"
)
read_gt3x(file_3x)
```

---

reintegrate	<i>Reintegrate a data stream</i>
-------------	----------------------------------

---

**Description**

Reintegrate a data stream

**Usage**

```
reintegrate(ag, to, time_var = "Timestamp", direction = c("forwards",
  "backwards"))
```

**Arguments**

ag	A data frame to reintegrate
to	The epoch length desired. Starting epoch length will be determined automatically.
time_var	The name of the column containing POSIX-formatted timestamp information
direction	The direction of reintegration, i.e. whether a timestamp refers to the timespan after the previous data point ("backwards"), or before the next data point ("forwards").

**Examples**

```
data("imu_to_check", package = "AGread")
ag <-
  imu_to_check[, c("Timestamp", "mean_abs_Gyroscope_x_DegPerS")]

# Forwards reintegration
reintegrate(
  ag = ag,
  to = 60,
  time_var = "Timestamp",
  direction = c("forwards")
)

# Backwards reintegration
reintegrate(
  ag = ag,
  to = 60,
  time_var = "Timestamp",
  direction = c("backwards")
)

## Not run:
# Erronious usages that will give a warning
reintegrate(
  ag = ag,
```

```
    to = 60,  
    time_var = "Timestamp",  
    direction = c("forwards", "backwards")  
  )  
  
  reintegrate(  
    ag = ag,  
    to = 60,  
    time_var = "Timestamp"  
  )  
  
## End(Not run)
```

# Index

## \*Topic **datasets**

- imu\_to\_check, [13](#)
- imu\_to\_collapse, [13](#)
- raw\_to\_collapse, [14](#)

- AG\_collapse, [3](#)
- AG\_meta, [4](#)
- AGread, [2](#)
- AGread-package (AGread), [2](#)

- check\_columns, [5](#), [17](#)
- check\_second, [5](#)
- classify\_magnetometer, [6](#)
- collapse\_gt3x, [7](#)

- g.calibrate, [17](#)
- get\_day\_of\_year, [8](#)
- get\_duration, [8](#)
- get\_imu\_file\_meta, [9](#)
- get\_minute, [9](#)
- get\_raw\_file\_meta, [10](#)
- get\_VM, [11](#)

- imu\_collapse, [11](#)
- imu\_filter\_gyroscope, [12](#)
- imu\_to\_check, [13](#)
- imu\_to\_collapse, [13](#)

- raw\_to\_collapse, [14](#)
- read.csv, [5](#)
- read\_AG\_counts, [2](#), [15](#)
- read\_AG\_IMU, [2](#), [16](#)
- read\_AG\_raw, [2](#), [17](#)
- read\_gt3x, [2](#), [7](#), [18](#)
- reintegrate, [19](#)