Package ‘GENEAcore’

April 9, 2025

Title Pre-Processing of 'GENEActiv' Data

Version 1.0.1

Date 2025-04-09

Maintainer Jia Ying Chua <jiayingc@activinsights.com>

Description
Analytics to read in and segment raw 'GENEActiv' accelerometer data into epochs and events. For more details on the 'GENEActiv' device, see <https://activinsights.com/resources/geneactiv-support-1-2/>.

License GPL (>= 2)

Encoding UTF-8

RoxygenNote 7.3.2

Imports changepoint, signal, methods

Suggests knitr, rmarkdown, testthat (>= 3.0.0), GENEAread (>= 2.0.10), GENEAclassify

Config/testthat/edition 3

VignetteBuilder knitr

NeedsCompilation no

Author Joss Langford [aut],
Ian Long [aut],
Jia Ying Chua [aut, cre],
Activinsights Ltd [cph]

Repository CRAN

Date/Publication 2025-04-09 09:30:09 UTC

Contents

`aggregateEpochs` *Aggregate Epochs*

Description

Aggregate Epochs

Usage

```
aggregateEpochs(
  time_series,
  measure = "AGSA",
  time = "timestamp",
  sample_frequency,
  duration = NA,
  first_epoch_timestamp = NA,
  fun = mean
)
```

Arguments

`time_series` Data frame to be aggregated.

`measure` Name of the measure columns to be included.

`time` Name of the time column.

`sample_frequency` Measurement frequency of data.

`duration` Time duration to aggregate in each epoch.

`first_epoch_timestamp`
Time to start the first epoch, defaults to first record.

`fun` Function to apply on aggregation, defaults to mean.

Details

Wrapper function that calls `aggregatePeriods` for epochs (duration of fixed length).

Value

Data frame of aggregated epochs.

Examples

```
timestamp <- c(
  1619424004, 1619424005, 1619424006, 1619424007,
  1619424008, 1619424009, 1619424010, 1619424011,
  1619424012, 1619424013, 1619424014, 1619424015
)
value <- c(
  0.729614366, 1.729115871, 0.804973546, 2.510181118,
  2.23764038, 0.613203747, 0.681953275, 0.089566943,
  0.021042388, 2.4780338, 2.437488989, 2.632635727
)
data <- data.frame(timestamp, value)
aggregated <- aggregateEpochs(data,
  duration = 5,
  measure = "value",
  sample_frequency = 1,
  first_epoch_timestamp = 1619424005,
  time = "timestamp"
)
```

`aggregateEvents` *Aggregate Events*

Description

Aggregate Events

Usage

```
aggregateEvents(
  time_series,
  measure = "AGSA",
  time = "timestamp",
  sample_frequency,
  events = NA,
  start_time = "start",
  end_time = "end",
  fun = mean
)
```

Arguments

time_series Data frame to be aggregated.

measure Name of the measure columns to be included.

time Name of the time column.

sample_frequency Measurement frequency of data.

events Data frame containing the start and end index of each event.

start_time Name of the column in events containing the start index of the events.

end_time Name of the column in events containing the end index of the events.

fun Function to apply on aggregation, defaults to mean.

Details

Wrapper function that calls aggregatePeriods for events (duration of variable length).

Value

Data frame of aggregated events.

Examples

```
timestamp <- c(
  1619424004, 1619424005, 1619424006, 1619424007,
  1619424008, 1619424009, 1619424010, 1619424011,
  1619424012, 1619424013, 1619424014, 1619424015
)
value <- c(
  0.729614366, 1.729115871, 0.804973546, 2.510181118,
  2.23764038, 0.613203747, 0.681953275, 0.089566943,
  0.021042388, 2.4780338, 2.437488989, 2.632635727
)
data <- data.frame(timestamp, value)
event_start <- c(1, 5, 10)
event_end <- c(4, 9, 12)
aggregated_events <- aggregateEvents(data,
  events = data.frame(start = event_start, end = event_end),
  measure = "value",
  time = "timestamp",
  start_time = "start",
  end_time = "end",
  sample_frequency = 1,
  fun = sum
)
```

aggregatePeriods *Aggregate Periods*

Description

Generalised aggregation function generates distinct epochs or events outputs based on the initial parameters provided.

Usage

```
aggregatePeriods(
  time_series,
  measure = "AGSA",
  time = "timestamp",
  sample_frequency,
  duration = NA,
  first_epoch_timestamp = NA,
  events = NA,
  start_time = "start",
  end_time = "end",
  fun = mean
)
```

Arguments

`time_series` Data frame to be aggregated.

`measure` Name of the measure columns to be included.

`time` Name of the time column.

`sample_frequency` Frequency of data.

`duration` Time duration to aggregate in each epoch.

`first_epoch_timestamp` Time to start the first epoch, defaults to first record.

`events` Data frame containing the start and end index of each event.

`start_time` Name of the column in events containing the start index of the events.

`end_time` Name of the column in events containing the end index of the events.

`fun` Function to apply on aggregation, defaults to mean.

Value

Data frame of aggregated epochs or events.

apply_AGSA *Apply Absolute Gravity-Subtracted Acceleration (AGSA)*

Description

Apply Absolute Gravity-Subtracted Acceleration (AGSA)

Usage

```
apply_AGSA(x)
```

Arguments

`x` Calibrated acceleration data frame.

Value

Measure column appended to end of calibrated data frame.

Examples

```
x <- c(0.14268, 0.21757, -0.529, -0.36383)
y <- c(0.26385, 0.27295, 0.29220, 0.79510)
z <- c(0.27722, 0.20296, 0.35092, 0.27459)
calibrated <- data.frame(x, y, z)
calibrated <- apply_AGSA(calibrated)
```

apply_calibration *Apply Calibration*

Description

Apply Calibration

Usage

```
apply_calibration(sensor_data, cal_params, measurement_device, use_temp = TRUE)
```

Arguments

`sensor_data` Raw sensor-level data from a bin file in the form (x, y, z, light, button, temp).

`cal_params` Calibration parameters for acceleration and light from MPI.

`measurement_device` Name of the measurement device used "GENEActiv 1.1" or "GENEActiv 1.2".

`use_temp` Allows auto-calibration to be run with and without temperature compensation.

Details

Function to apply calibration to sensor-level data from a bin file.

Value

Data frame of calibrated sensor data.

Examples

```
cal_params <- list(
  scale = c(1.015, 1.017, 1.027),
  offset = c(0.00128, 0.0383, 0.0138),
  temperatureoffset = c(0, 0, 0),
  error = NA,
  lightdenominator = 48,
  lightnumerator = 911
)

rawdata <- data.frame(
  time = c(rep(1726650857, 5)),
  x = c(0.2421875,0.24609375,0.25390625,0.24609375,0.23828125),
  y = c(-0.04296875,-0.04687500,-0.03515625, -0.03125000,-0.04296875),
  z = c(-0.9453125,-0.9453125, -0.9531250,-0.9531250,-0.9609375),
  light = c(rep(22, 5)),
  button = c(rep(0, 5)),
  temp = c(rep(21.3, 5)),
  volts = c(rep(4.0896, 5))
)
calibrated <- apply_calibration(rawdata, cal_params, "GENEActiv 1.1")
```

`apply_degrees` *Apply Rotation (degrees)*

Description

Apply Rotation (degrees)

Usage

```
apply_degrees(x)
```

Arguments

`x` Calibrated acceleration data frame.

Value

Measure column appended to end of calibrated data frame.

Examples

```
x <- c(0.14268, 0.21757, -0.529, -0.36383)
y <- c(0.26385, 0.27295, 0.29220, 0.79510)
z <- c(0.27722, 0.20296, 0.35092, 0.27459)
calibrated <- data.frame(x, y, z)
calibrated <- apply_degrees(calibrated)
```

---

apply_ENMO *Apply Euclidean Norm Minus One (ENMO)*

Description

Apply Euclidean Norm Minus One (ENMO)

Usage

```
apply_ENMO(x)
```

Arguments

x	Calibrated acceleration data frame.

Value

Measure column appended to end of calibrated data frame.

Examples

```
x <- c(0.14268, 0.21757, -0.529, -0.36383)
y <- c(0.26385, 0.27295, 0.29220, 0.79510)
z <- c(0.27722, 0.20296, 0.35092, 0.27459)
calibrated <- data.frame(x, y, z)
calibrated <- apply_ENMO(calibrated)
```

---

apply_radians *Apply Rotation (radians)*

Description

Apply Rotation (radians)

Usage

```
apply_radians(x)
```

Arguments

`x`	Calibrated acceleration data frame.

Value

Measure column appended to end of calibrated data frame.

Examples

```
x <- c(0.14268, 0.21757, -0.529, -0.36383)
y <- c(0.26385, 0.27295, 0.29220, 0.79510)
z <- c(0.27722, 0.20296, 0.35092, 0.27459)
calibrated <- data.frame(x, y, z)
calibrated <- apply_radians(calibrated)
```

`apply_updown` *Apply Elevation (updown)*

Description

Apply Elevation (updown)

Usage

```
apply_updown(x)
```

Arguments

`x`	Calibrated acceleration data frame.

Value

Measure column appended to end of calibrated data frame.

Examples

```
x <- c(0.14268, 0.21757, -0.529, -0.36383)
y <- c(0.26385, 0.27295, 0.29220, 0.79510)
z <- c(0.27722, 0.20296, 0.35092, 0.27459)
calibrated <- data.frame(x, y, z)
calibrated <- apply_updown(calibrated)
```

binfile_summary *Bin File Summary*

Description

Bin File Summary

Usage

```
binfile_summary(input, recursive = TRUE)
```

Arguments

input	Bin file path.
recursive	TRUE applies the operation to all nested elements.

Details

Wrapper function that calls `create_summary` for bin files only.

Value

Data frame of bin file or MPI summary.

calc_autocalparams *Calculate Auto-calibration Parameters*

Description

Function to calculate auto-calibration parameters from known still points from a bin file that create a unitary sphere.

Usage

```
calc_autocalparams(
  binfile,
  binfile_path,
  output_folder,
  sphere_points,
  use_temp = TRUE,
  spherecrit = 0.3,
  maxiter = 500,
  tol = 1e-13
)
```

Arguments

binfile Text lines read from an open connection to a bin file.

binfile_path Path to the bin file to be processed.

output_folder Path to the folder containing GENEAcore run outputs and Measurement Period Information (MPI) files.

sphere_points List of points that populate a unitary sphere and their associated temperature in the form (x,y,z,temp).

use_temp Allows auto-calibration to be run with and without temperature compensation.

spherecrit The minimum required acceleration value for each axis in both directions for auto-calibration to be reliable.

maxiter The maximum number of sphere fit iterations attempted during auto-calibration.

tol The limit of incremental sphere fit improvements before auto-calibration is considered complete.

Value

List of auto-calibration parameters within the measurement period information (MPI).

Examples

```
binfile_path <- system.file("inst/extdata/10Hz_calibration_file.bin", package = "GENEAcore")
output_folder <- "."
con <- file(binfile_path, "r")
binfile <- readLines(con, skipNul = TRUE)
close(con)
MPI <- create_MPI(binfile, binfile_path, output_folder)
nonmovement_list <- detect_nonmovement(binfile, binfile_path, output_folder)
MPI <- calc_autocalparams(binfile, binfile_path, output_folder, nonmovement_list$sphere_points)
```

createEventMapping *Create Event Mapping*

Description

Create Event Mapping

Usage

```
createEventMapping(events, start_time, end_time, max_row_number)
```

Arguments

events Data frame containing the start and end index of each event.

start_time Name of the column in events containing the start index of the events.

end_time Name of the column in events containing the end index of the events.

max_row_number Number of rows in the source vector the events describe

Details

Enumerate a vector to identify which event each measurement belongs to.

Value

List of mapped events.

Examples

```
events <- data.frame(
  "start" = c(1, 5, 10, 15),
  "end" = c(4, 9, 14, 19)
)
time_series <- rnorm(25)
period_number <- createEventMapping(events, "start", "end", length(time_series))
```

---

create_MPI *Create Measurement Period Information*

Description

Create Measurement Period Information

Usage

```
create_MPI(binfile, binfile_path, output_folder, out_rds = TRUE)
```

Arguments

binfile	Text lines read from an open connection to a bin file.
binfile_path	Path to the bin file to be processed.
output_folder	Folder to write MPI file in.
out_rds	Allows RDS output to be saved during MPI creation.

Details

Function to create measurement period information (MPI) from a GENEActiv bin file

Value

List of measurement period information.

Examples

```
binfile_path <- system.file("inst/extdata/20Hz_file.bin", package = "GENEAcore")
con <- file(binfile_path, "r")
binfile <- readLines(con, skipNul = TRUE)
close(con)
MPI <- create_MPI(binfile)
```

`create_summary` *Create Summary*

Description

Create Summary

Usage

```
create_summary(input, path_type, recursive)
```

Arguments

`input`	Input type of either a bin file path, MPI path or an MPI object.
`path_type`	The file type within the folder to create summary for.
`recursive`	TRUE applies the operation to all nested elements.

Details

Function to create a summary of key information of a bin file or MPI path.

Value

Data frame of bin file or MPI summary.

`detect_nonmovement` *Detect Non-movement*

Description

Detect Non-movement

Usage

```
detect_nonmovement(
  binfile,
  binfile_path,
  output_folder,
  still_seconds = 120,
  sd_threshold = 0.011,
  temp_seconds = 240,
  border_seconds = 300,
  long_still_seconds = 120 * 60,
  delta_temp_threshold = -0.7,
  posture_changes_max = 2,
  non_move_duration_max = 12 * 60 * 60
)
```

Arguments

`binfile` Text lines read from an open connection to a bin file.

`binfile_path` Path to the bin file to be processed.

`output_folder` Path to the folder containing GENEAcore run outputs and Measurement Period Information (MPI) files.

`still_seconds` The number of seconds included in the rolling standard deviation calculation for stillness to determine the shortest detection duration.

`sd_threshold` The threshold applied to the rolling standard deviation of combined acceleration to determine stillness.

`temp_seconds` The number of seconds included in the rolling temperature difference calculation or non-wear which also determines the shortest detection duration.

`border_seconds` The minimum number of seconds of a still event to be classed as a new bout.

`long_still_seconds` The minimum number of seconds of a still bout that is classed as non-wear.

`delta_temp_threshold` The threshold applied to the rolling temperature difference to determine non-wear.

`posture_changes_max` The maximum number of adjoining events that make up a single bout.

`non_move_duration_max` The maximum number of seconds of a still bout to be classed as non-movement. Still bouts with a duration longer than this number is automatically classed as non-wear.

Details

Function to detect non-movement bouts, non-wear events and points in a 1Hz downsampled bin file.

Value

List of sphere points, non-movement bouts and non-wear events.

Examples

```
binfile_path <- system.file("inst/extdata/20Hz_file.bin", package = "GENEAcore")
con <- file(binfile_path, "r")
binfile <- readLines(con, skipNul = TRUE)
close(con)
output_folder <- "."
MPI <- create_MPI(binfile, binfile_path, output_folder)
MPI <- detect_nonmovement(binfile, binfile_path, output_folder)
```

detect_transitions *Detect Transitions*

Description

Detect Transitions

Usage

```
detect_transitions(
  binfile,
  binfile_path,
  output_folder,
  minimum_event_duration = 3,
  x_cpt_penalty = 20,
  y_cpt_penalty = 30,
  z_cpt_penalty = 20,
  CutTime24Hr = "15:00"
)
```

Arguments

`binfile` Text lines read from an open connection to a bin file.

`binfile_path` Path to the bin file to be processed.

`output_folder` Path to the folder containing GENEAcore run outputs and Measurement Period Information (MPI) files.

`minimum_event_duration` The minimum interval between changepoint transitions.

`x_cpt_penalty` The manual penalty value applied in the PELT changepoint algorithm for the x axis, see `cpt.var`.

`y_cpt_penalty` The manual penalty value applied in the PELT changepoint algorithm for the y axis, see `cpt.var`.

`z_cpt_penalty` The manual penalty value applied in the PELT changepoint algorithm for the z axis, see `cpt.var`.

`CutTime24Hr` Time in 24h to split the days up by.

Details

Function to detect mean and variance changepoints in 1Hz acceleration data from a bin file.

Value

List of time, index and day number of each transition within the measurement period information.

Examples

```
binfile_path <- system.file("inst/extdata/20Hz_file.bin", package = "GENEAcore")
con <- file(binfile_path, "r")
binfile <- readLines(con, skipNul = TRUE)
close(con)
output_folder <- "."
MPI <- create_MPI(binfile, binfile_path, output_folder)
MPI <- detect_transitions(binfile, binfile_path, output_folder)
```

geneacore *Main GENEAcore Function*

Description

Main GENEAcore Function

Usage

```
geneacore(
  data_folder = data_folder,
  CutTime24Hr = "15:00",
  output_epochs = TRUE,
  epoch_duration = 1,
  output_events = TRUE,
  output_steps = FALSE,
  output_csv = FALSE,
  timer = FALSE
)
```

Arguments

`data_folder`	Folder that contains raw data bin files to process.
`CutTime24Hr`	Time in 24h to split the days up by.
`output_epochs`	Create epoch outputs.
`epoch_duration`	Specify duration of fixed epochs.
`output_events`	Create event outputs.
`output_steps`	Include step counts and stepping rate outputs.
`output_csv`	Allows CSV output to be saved during epoch and event processing.
`timer`	Print elapsed times of each process.

Value

RDS and CSV of Measurement Period Information, Epoch measures and Event measures.

get_decimal_places *Get Decimal Places*

Description

Get Decimal Places

Usage

```
get_decimal_places(column)
```

Arguments

column	Aggregated data frame column.

Details

Function to determine the number of decimal places based on column name.

Value

Decimal place integer.

Examples

```
epochs_df <- data.frame(
  "x.mean" = c(0.1111, 0.1222, 0.1333, 0.1444),
  "y.mean" = c(0.2111, 0.2222, 0.2333, 0.2444),
  "light.mean" = c(1.25, 1.73, 1.99, 2.02)
)
dp <- get_decimal_places(epochs_df[1])
```

get_UniqueBinFileIdentifier *Generate Unique Bin File Identifier*

Description

Generate Unique Bin File Identifier

Usage

```
get_UniqueBinFileIdentifier(binfile)
```

Arguments

binfile	Text lines read from an open connection to a bin file.

Details

Function to create a UniqueBinFileIdentifier from a GENEActiv bin file.

Value

Single string identifier.

Examples

```
binfile_path <- system.file("inst/extdata/20Hz_file.bin", package = "GENEAcore")
con <- file(binfile_path, "r")
binfile <- readLines(con, skipNul = TRUE)
close(con)
UniqueBinFileIdentifier <- get_UniqueBinFileIdentifier(binfile)
```

MPI_summary *MPI Summary*

Description

MPI Summary

Usage

```
MPI_summary(input, recursive = TRUE)
```

Arguments

`input`	MPI path.
`recursive`	TRUE applies the operation to all nested elements.

Details

Wrapper function that calls `create_summary` for MPI only.

Value

Data frame of MPI summary.

new_cut_times *New Cut Times*

Description

New Cut Times

Usage

```
new_cut_times(df)
```

Arguments

df	Cut Times data frame.

Details

Add the timestamps, indexes and day numbers of the cut times and their ends.

Value

Data frame with added cut times.

Examples

```
CutTimes_df <- data.frame(
  time = c(1731421000, 1731421100, 1731421362, 1731421480, 1731421525),
  index = c(56, 1, 230, 1, 400), day = c(1, 2, 2, 3, 3)
)
CutTimes_df <- new_cut_times(CutTimes_df)
```

sample_binfile *Sample Bin File*

Description

Sample Bin File

Usage

```
sample_binfile(
  binfile,
  binfile_path,
  output_folder,
  start_time = NULL,
  end_time = NULL,
  downsample = TRUE,
  output_csv = FALSE
)
```

Arguments

`binfile` Text lines read from an open connection to a bin file.

`binfile_path` Path to the bin file to be processed.

`output_folder` Path to the folder containing GENEAcore run outputs and Measurement Period Information (MPI) files.

`start_time` Time stamp to start the read from, default start of file.

`end_time` Time stamp to end the read from, default end of file.

`downsample` Logical to determine whether to downsample the file, default TRUE.

`output_csv` Allow outputs of bin file sampling to be saved as CSV.

Details

Function to read in a GENEActiv bin file with option to downsample to 1Hz.

Value

List of 1Hz downsampled data or raw sample data.

Examples

```
binfile_path <- system.file("inst/extdata/20Hz_file.bin", package = "GENEAcore")
output_folder <- "."
con <- file(binfile_path, "r")
binfile <- readLines(con, skipNul = TRUE)
close(con)
measurements <- sample_binfile(binfile, binfile_path, output_folder)
```

`stepCounter` *Step Counter*

Description

Function to calculate the number and variance of the steps in the data.

Usage

```
stepCounter(
  StepData,
  samplefreq = 100,
  filterorder = 2,
  boundaries = c(0.5, 5),
  Rp = 3,
  hysteresis = 0.05,
  fun = c("GENEAcount", "mean", "sd")
)
```

Arguments

`StepData` The data to use for calculating the steps. This should either an AccData object or a vector.

`samplefreq` The sampling frequency of the data, in hertz, when calculating the step number (default 100).

`filterorder` single integer, order of the Chebyshev bandpass filter, passed to argument n of `cheby1`.

`boundaries` length 2 numeric vector specifying lower and upper bounds of Chebychev filter (default `c(0.5, 5)` Hz), passed to argument W of `butter` or `cheby1`.

`Rp` the decibel level that the cheby filter takes, see `cheby1`.

`hysteresis` The hysteresis applied after zero crossing. (default 100mg)

`fun` character vector naming functions by which to summarize steps. "count" is an internally implemented summarizing function that returns step count.

Value

Returns a vector with length fun.

Examples

```
d1 <- sin(seq(0.1, 100, 0.1))/2 + rnorm(1000)/10 + 1
Steps4 = stepCounter(d1)
length(Steps4)
mean(Steps4)
sd(Steps4)
plot(Steps4)
```

Index