Package 'GENEAcore'

April 2, 2026

Title Pre-Processing of 'GENEActiv' Data

Version 1.2.0

Date 2026-03-30

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, jsonlite

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

Config/testthat/edition 3

VignetteBuilder knitr

NeedsCompilation no

Depends R (>= 3.5.0)

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

Repository CRAN

Date/Publication 2026-04-02 13:40:02 UTC

Contents

`aggregate_epochs` *Aggregate Epochs*

Description

Aggregate Epochs

Usage

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

aggregateEpochs(...)
```

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.
...	Additional arguments passed to internal aggregation functions.

Details

Wrapper function that calls `aggregate_periods` 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 <- aggregate_epochs(data,
  duration = 5,
  measure = "value",
  sample_frequency = 1,
  first_epoch_timestamp = 1619424005,
  time = "timestamp"
)
```

aggregate_events *Aggregate Events*

Description

Aggregate Events

Usage

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

aggregateEvents(...)

aggregatePeriods(...)

createEventMapping(...)
```

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.
...	Additional arguments passed to internal aggregation functions.

Details

Wrapper function that calls aggregate_periods 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 <- aggregate_events(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
)
```

---

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", "GENEActiv 1.2" or "GENEActiv 1.3".
`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),
  temperature_offset = c(0, 0, 0),
  error = NA,
  light_denominator = 48,
  light_numerator = 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, identifier_mapping_record = NULL)
```

Arguments

input	Bin file path - single bin file or folder of bin files.
recursive	TRUE applies the operation to all nested elements.
identifier_mapping_record	File path of the exported identifier_mapping_record CSV file.

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 to ensure sufficient range of non-movement positions for auto-calibration to be reliable.
`maxiter`	The maximum number of sphere fit iterations attempted during auto-calibration to converge.
`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("extdata/10Hz_calibration_file_20Nov25.bin", package = "GENEAcore")
output_folder <- tempdir()
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$non_movement$sphere_points
)
```

---

check_MPI *Print out a summary of MPI file(s)*

Description

Function to print out a summary of an MPI (Measurement Process Information) file, or all MPI files found if a directory path is supplied.

Usage

```
check_MPI(input, print_output = TRUE)
```

Arguments

input	Fully qualified path to a MPI file or a directory containing MPI files, the directory is searched recursively.
print_output	Boolean to print MPI summary to console.

Examples

```
## Not run:
check_MPI("fully qualified path to a directory structure containing MPI files")
check_MPI("fully qualified path to a MPI file")

## End(Not run)
```

create_coel_atoms_json *Create COEL Behavioural Atoms JSON*

Description

Creates per-participant rest activity and behavioural bout atoms and a payload

Usage

```
create_coel_atoms_json(data_folder)
```

Arguments

data_folder	Folder that contains raw data bin files for which bouts have been created for.

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("extdata/10Hz_calibration_file_20Nov25.bin", package = "GENEAcore")
con <- file(binfile_path, "r")
binfile <- readLines(con, skipNul = TRUE)
close(con)
output_folder <- tempdir()
MPI <- create_MPI(binfile, binfile_path, output_folder, out_rds = FALSE)
```

`detect_nonmovement` *Detect Non-movement*

Description

Detect Non-movement

Usage

```
detect_nonmovement(
  binfile,
  binfile_path,
  output_folder,
  still_seconds = 120,
  sd_threshold = 0.013,
  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 of non-movement from 1Hz downsampled data.
`sd_threshold`	The threshold applied to the rolling standard deviation of mean acceleration standard deviation of 1Hz downsampled data to determine non-movement.
`temp_seconds`	The number of seconds included in the rolling temperature difference calculation for non-wear, which also determines the shortest detection duration.
`border_seconds`	The maximum number of seconds between non-movement events for them to be combined into the same period.

`long_still_seconds`
: The number of seconds for any single non-movement event beyond which the whole period 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 non-movement events that make up a single period of non-wear less than the maximum non-move duration.

`non_move_duration_max`
: The number of seconds beyond which non-movement events are automatically classed as non-wear.

Details

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

Value

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

Examples

```
binfile_path <- system.file("extdata/10Hz_calibration_file_20Nov25.bin", package = "GENEAcore")
con <- file(binfile_path, "r")
binfile <- readLines(con, skipNul = TRUE)
close(con)
output_folder <- tempdir()
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(
  measurements,
  minimum_event_duration = 5,
  x_cpt_penalty = 18,
  y_cpt_penalty = 25,
  z_cpt_penalty = 16,
  cut_time_24hr = "15:00"
)
```

Arguments

measurements	The downsampled measurements to calculate changepoint transitions.
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.
cut_time_24hr	Time in 24h to split days up by.

Details

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

Value

List of time and index of each transition.

---

folder_measures *Daily Measures for Data Folder*

Description

Daily Measures for Data Folder

Usage

```
folder_measures(
  data_folder,
  measure_type = c("activity", "sleep", "both"),
  minimum_valid_hours = 22,
  include_participant_info = FALSE,
  identifier_mapping_record = NULL
)
```

Arguments

data_folder	Folder that contains raw data bin files to process or path to single bin file.
measure_type	The type of daily measure summary to generate.
minimum_valid_hours	Minimum hours of wear time in a 24-hour day to be considered a valid day for reported measures.

include_participant_info
: If TRUE, appends participant information extracted from the bin file to both the activity and sleep measures.

identifier_mapping_record
: File path of the exported identifier_mapping_record CSV file.

Details

Wrapper to process and output daily activity measures aggregate for all bin files in the data folder

Value

Daily aggregated activity measures (.csv)

---

geneabout *GENEAbout*

Description

GENEAbout function which performs the following tasks for each available day:

- Assign bout classifications to events only
- Reports aggregated daily activity and sleep measures
- Creates COEL v2.0 Behavioural Atoms JSON

Usage

```
geneabout(data_folder = data_folder, control = list())
```

Arguments

data_folder	Folder that contains raw data bin files to process or path to single bin file.
control	Named list of optional settings for geneacore functions.

Details

Control options (all optional):

timer Logical. Default FALSE. Print elapsed times of each processing step.

minimum_valid_hours Integer. Default 22. Minimum hours of wear time in a 24-hour day to be considered a valid day for reported measures.

save_daily_bouts Logical. Default FALSE. If TRUE,daily bouts are saved as RDS files and CSV files (if output_csv is set to TRUE). If FALSE, only a single RDS file containing bouts for all processed days is saved.

coel_json Logical. Default FALSE. Exports bouts as COEL behavioural atoms in JSON format. For each bin file in the data folder, behavioural bout atoms and rest activity atoms are produced. Additionally, aggregated JSON files are created in the Batch JSON outputs subfolder.

identifier_mapping_record String. Default NULL. If a file path to the exported identifier_mapping_record.csv is provided, encrypted participant IDs in the activity and sleep measures will be replaced with their corresponding mapped identifiers.

Examples

```
## Not run:
controls <- list(minimum_valid_hours = 4, timer = TRUE, coel_json = TRUE)
geneabout("path/to/folder", controls)

## End(Not run)
```

---

geneacore *GENEAcore Main Wrapper*

Description

GENEAcore function that wraps the functionalities of geneacore_part1 to geneacore_part4. GENEAcore function which performs the following tasks for each valid day:

- Checks and reads the data file and creates the Measurement Period Information (MPI)
- Downsamples the file to 1Hz and detects periods of non movement
- Calculates auto calibration parameters for the device
- Samples daily raw bin file data and saves as daily RDS files
- Reads and saves button press timestamps to the Measurement Period Information (MPI)
- Applies calibration parameters to raw data
- Identifies transitions for events processing only
- Performs aggregation by event and/or epoch
- Calculate steps, if option selected
- Calculate non-wear coverage and rest coverage of events and/or epochs
- Assign bout classifications to events only
- Reports aggregated daily activity and sleep measures
- Consolidates daily epochs or bouts outputs into a single file

Usage

```
geneacore(data_folder = data_folder, control = list())
```

Arguments

data_folder	Folder that contains raw data bin files to process or path to single bin file.
control	Named list of optional settings for geneacore functions.

Details

Control options (all optional):

cut_time_24hr String. Default "15:00". Time in 24h to split days up by.

output_epochs Logical. Default `FALSE`. Create epoch outputs.

epoch_duration Integer. Default 1. Specify duration of fixed epochs.

output_events Logical. Default `TRUE`. Create event outputs.

output_steps Logical. Default `FALSE`. Calculate step counts and stepping rate during epoch processing. Steps are always calculated during events processing.

output_csv Logical. Default `FALSE`. Allows CSV output to be saved during epoch and event processing.

timer Logical. Default `FALSE`. Print elapsed times of each processing step.

multisession Integer vector of length 2. Default `c(1,1)`. Controls day split when running function across multiple R sessions. The first value is the current session number and the second is the total number of sessions. The default processes all days in a single session.

required_processing_hours Integer. Default 0. Hours of wear time in a 24-hour day to be considered a valid day to be processed.

Examples

```
## Not run:
controls <- list(output_csv = TRUE, required_processing_hours = 4, timer = TRUE)
geneacore("path/to/folder", controls)

## End(Not run)
```

`geneacore_part1` *GENEAcore Part 1*

Description

GENEAcore function which performs the following tasks:

- Checks and reads the data file and creates the Measurement Period Information (MPI)
- Downsamples the file to 1Hz and detects periods of non movement
- Calculates auto calibration parameters for the device

Usage

```
geneacore_part1(data_folder = data_folder, control = list())
```

Arguments

`data_folder`	Folder that contains raw data bin files to process or path to single bin file.
`control`	Named list of optional settings for geneacore functions.

Details

Control options (all optional):

cut_time_24hr String. Default "15:00". Time in 24h to split days up by.

timer Logical. Default `FALSE`. Print elapsed times of each processing step.

Value

Measurement Period Information (.rds, .json) and 1Hz downsampled data (.rds)

Examples

```
## Not run:
controls <- list(output_csv = TRUE, required_processing_hours = 4, timer = TRUE)
geneacore_part1("path/to/folder", controls)

## End(Not run)
```

`geneacore_part2` *GENEAcore Part 2*

Description

GENEAcore function which performs the following tasks for each valid day:

- Samples daily raw bin file data and saves as daily RDS files
- Reads and saves button press timestamps to the Measurement Period Information (MPI)

Usage

```
geneacore_part2(data_folder = data_folder, control = list())
```

Arguments

`data_folder`	Folder that contains raw data bin files to process or path to single bin file.
`control`	Named list of optional settings for geneacore functions.

Details

Control options (all optional):

required_processing_hours Integer. Default 0. Hours of wear time in a 24-hour day to be considered a valid day to be processed.

timer Logical. Default `FALSE`. Print elapsed times of each processing step.

Value

Daily raw data sample of x, y, z, Light, Button, Temperature and Voltage (.rds)

Examples

```
## Not run:
controls <- list(output_csv = TRUE, required_processing_hours = 4, timer = TRUE)
geneacore_part2("path/to/folder", controls)

## End(Not run)
```

---

geneacore_part3 *GENEAcore Part 3*

Description

GENEAcore function which performs the following tasks for each valid day:

- Applies calibration parameters to raw data
- Identifies transitions for events processing only
- Performs aggregation by event and/or epoch
- Calculate steps, if option selected
- Calculate non-wear coverage and rest coverage of events and/or epochs

Usage

```
geneacore_part3(data_folder = data_folder, control = list())
```

Arguments

data_folder	Folder that contains raw data bin files to process or path to single bin file.
control	Named list of optional settings for geneacore functions.

Details

Control options (all optional):

output_epochs Logical. Default FALSE. Create epoch outputs.

epoch_duration Integer. Default 1. Specify duration of fixed epochs.

output_events Logical. Default TRUE. Create event outputs.

output_steps Logical. Default FALSE. Calculate step counts and stepping rate during epoch processing. Steps are always calculated during events processing.

output_csv Logical. Default FALSE. Allows CSV output to be saved during epoch and event processing.

timer Logical. Default FALSE. Print elapsed times of each processing step.

multisession Integer vector of length 2. Default c(1,1). Controls day split when running function across multiple R sessions. The first value is the current session number and the second is the total number of sessions. The default processes all days in a single session.

required_processing_hours Integer. Default 0. Hours of wear time in a 24-hour day to be considered a valid day to be processed.

Examples

```
## Not run:
controls <- list(output_csv = TRUE, required_processing_hours = 4, timer = TRUE)
geneacore_part3("path/to/folder", controls)

## End(Not run)
```

---

geneacore_part4 *GENEAcore Part 4*

Description

GENEAcore function which performs the following tasks:

- Consolidates daily epochs or bouts outputs into a single file

Usage

```
geneacore_part4(data_folder = data_folder, control = list())
```

Arguments

data_folder	Folder that contains raw data bin files to process or path to single bin file.
control	Named list of optional settings for geneacore functions.

Details

Control options (all optional):

output_epochs Logical. Default TRUE. Create epoch outputs.

epoch_duration Integer. Default 1. Specify duration of fixed epochs.

output_events Logical. Default TRUE. Create event outputs.

output_csv Logical. Default FALSE. Allows CSV output to be saved during epoch and event processing.

Value

Consolidated epochs and/or bouts output (.rds/.csv)

Examples

```
## Not run:
controls <- list(output_csv = TRUE, minimum_valid_hours = 4, timer = TRUE)
geneacore_part4("path/to/folder", controls)

## End(Not run)
```

`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("extdata/10Hz_calibration_file_20Nov25.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, identifier_mapping_record = NULL)
```

Arguments

input	MPI path - single MPI file or folder of MPI files.
recursive	TRUE applies the operation to all nested elements.
identifier_mapping_record	File path of the exported identifier_mapping_record CSV file.

Details

Wrapper function that calls `create_summary` for MPI only.

Value

Data frame of MPI summary.

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,
  save_raw = 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 data read, default start of file.
end_time	Time stamp to end the data read, 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.
save_raw	Save raw sampled data as RDS for quicker reprocessing, default FALSE.

Details

Function to read in a GENEActiv bin file with option to downsample to 1Hz. This can read the whole of the file or just a portion of it by setting the start_time and end_time parameters.

Value

List of 1Hz downsampled data or raw sample data.

Examples

```
binfile_path <- system.file("extdata/10Hz_calibration_file_20Nov25.bin", package = "GENEAcore")
output_folder <- tempdir()
con <- file(binfile_path, "r")
binfile <- readLines(con, skipNul = TRUE)
close(con)
measurements <- sample_binfile(binfile, binfile_path, output_folder)
```

---

step_counter *Step Counter*

Description

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

Usage

```
step_counter(
  step_data,
  sample_frequency = 100,
  filter_order = 2,
  boundaries = c(0.5, 5),
  Rp = 3,
  step_hysteresis = 0.05,
```

```
  fun = c("GENEAcount", "mean", "sd", "stepdiff")
)

stepCounter(...)
```

Arguments

step_data	The data to use for calculating the steps. This should be a vector of acceleration values.
sample_frequency	The sampling frequency of the data, in hertz, when calculating the step number (default 100).
filter_order	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.
step_hysteresis	The hysteresis applied after zero crossing of the bandpass filtered y-axis signal.
fun	character vector naming functions by which to summarize steps. "count" is an internally implemented summarizing function that returns step count.
...	Additional arguments passed to internal aggregation functions.

Value

Returns a vector with length fun.

Examples

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

Index