Package 'actimetrics'

September 17, 2026

Type Package

Title Create Metrics Actigraphy and Activity Analysis

Version 0.4.0

Description Provides functions for calibrating, counting, and summarizing actigraphy and activity data into specific metrics and sleep measures. The metrics include activity counts, step counts, activity index, Monitor Independent Movement Summary Unit (MIMS), mean amplitude deviation (MAD), and provides wrappers for sleep estimation from activity counts using Tudor-Locke (2014) <doi:10.1139/apnm-2013-0173> and Sadeh (1994) <doi:10.1093/sleep/17.3.201>.

License GPL-3

Depends R (>= 4.1.0)

Suggests testthat, utils, covr, knitr, httr, rmarkdown, MIMSunit, data.table, reticulate, stepcount (>= 0.6.0), actigraph.sleepr (>= 0.3.1), agcounts (>= 0.6.7), callr

Encoding UTF-8

VignetteBuilder knitr

Imports actibase, actiread, dplyr, lubridate, assertthat, purrr, tibble, rlang, stats, tidyr, walking (>= 0.8.0), methods

LazyData true

Config/roxygen2/version 8.0.0

URL https://jhuwit.github.io/actimetrics/

BugReports https://github.com/jhuwit/actimetrics/issues

NeedsCompilation no

Author John Muschelli [aut, cre] (ORCID: <https://orcid.org/0000-0001-6469-1750>)

Maintainer John Muschelli <muschellij2@gmail.com>

Repository CRAN

Date/Publication 2026-09-17 09:10:15 UTC

Contents

`acti_calculate_counts` *Process Count Data*

Description

Process Count Data

Process Count Data

Usage

```
acti_calculate_counts(
  data,
  epoch = 60L,
  resample = TRUE,
  lfe_select = FALSE,
  verbose = TRUE
)

acti_calculate_wear(
  data,
  method = c("choi", "troiano"),
  use_magnitude = TRUE,
  ...
)

acti_calculate_nonwear(
  data,
  method = c("choi", "troiano"),
  use_magnitude = TRUE,
  ...
```

```
)

acti_apply_cole_kripke(data)

acti_apply_tudor_locke(data, ...)

acti_apply_sadeh(data, ...)
```

Arguments

data	A data.frame from acti_calculate_counts that has columns axis1-3 and counts
epoch	epoch length in seconds. Default is 60 seconds. See agcounts::calculate_counts()
resample	(recommended) resample the data to 30Hz using actibase::acti_resample vs. using the resampling method from agcounts::calculate_counts.
lfe_select	Apply the Actigraph Low Frequency Extension filter. See agcounts::calculate_counts() higher values are higher levels of verbosity.
verbose	print diagnostic messages. Either logical or integer, where
method	Method for detecting non-wear, either "choi" or "troiano", corresponding to actigraph.sleepr::apply_choi() or actigraph.sleepr::apply_troiano()
use_magnitude	If TRUE, the magnitude of the vector (axis1, axis2, axis3) is used to measure activity; otherwise the axis1 value is used.
...	additional arguments to pass to actigraph.sleepr function

Value

A data.frame of transformed data

A data.frame of transformed data with columns axis1-3, counts, and counts_log10.

A data.frame of transformed data

Note

This calls the downstream wear-processing helpers used by actigraph.sleepr

Examples

```
path = actiread::acti_example_gt3x()
ac = actiread::acti_read_gt3x(path)
out = acti_calculate_counts(ac)

data = actimetrics::acti_count_data
wear = actimetrics::acti_calculate_wear(data)
tro_wear = actimetrics::acti_calculate_wear(data, method = "troiano")
ck = actimetrics::acti_apply_cole_kripke(data)
tl = actimetrics::acti_apply_tudor_locke(ck)
sadeh = actimetrics::acti_apply_sadeh(ck)
```

acti_calculate_forest *Calculate Step Counts via Oak/Forest*

Description

Calculate Step Counts via Oak/Forest

Usage

```
acti_calculate_forest(data, ...)
```

Arguments

data	A data.frame, AccData object, or GT3X file with X, Y, Z, and time
...	Additional arguments passed to walking::find_walking()

Value

A tibble with minute-level time, steps columns.

Examples

```
Sys.setenv("SSQ_PARALLEL" = 0)
if (reticulate::py_module_available("forest")) {
  data = actiread::acti_read_gt3x(actiread::acti_example_gt3x())
  steps = acti_calculate_forest(data, sample_rate = 100)
}
```

acti_calculate_measures *Calculate Summary Measures from Raw Accelerometer Data*

Description

Calculate Summary Measures from Raw Accelerometer Data

Usage

```
acti_calculate_measures(
  data,
  unit = "1 min",
  fix_zeros = TRUE,
  dynamic_range = NULL,
  calculate_mims = TRUE,
  calculate_ac = TRUE,
```

```
  flag_data = TRUE,
  flags = NULL,
  ensure_all_time = TRUE,
  verbose = TRUE,
  sample_rate = NULL,
  ...
)

acti_calculate_ai(
  data,
  unit = "1 min",
  ensure_all_time = TRUE,
  verbose = FALSE
)

acti_calculate_activity_index(
  data,
  unit = "1 min",
  ensure_all_time = TRUE,
  verbose = FALSE
)

acti_calculate_flags(data, unit = "1 min", ensure_all_time = TRUE)

acti_calculate_n_idle(data, unit = "1 min", ensure_all_time = TRUE)

acti_calculate_enmo(...)

acti_calculate_ai_defined(...)

acti_calculate_mad(
  data,
  unit = "1 min",
  ensure_all_time = TRUE,
  verbose = FALSE
)

acti_calculate_auc(
  data,
  unit = "1 min",
  sample_rate = NULL,
  allow_truncation = FALSE,
  ensure_all_time = TRUE,
  verbose = TRUE
)

acti_calculate_fast_mims(
  data,
```

```
  unit = "1 min",
  dynamic_range = NULL,
  sample_rate = NULL,
  allow_truncation = TRUE,
  ensure_all_time = TRUE,
  verbose = TRUE,
  ...
)

acti_calculate_mims(
  data,
  unit = "1 min",
  dynamic_range = c(-6, 6),
  ensure_all_time = TRUE,
  ...
)
```

Arguments

data	An object with columns X, Y, and Z or an object of class AccData
unit	length of time to calculate measures over. a character string specifying a time unit or a multiple of a unit to be rounded to. Valid base units are second, minute, hour, day, week, month, bimonth, quarter, season, halfyear, and year. Arbitrary unique English abbreviations as in the lubridate::period() constructor are allowed.
fix_zeros	Should actibase::acti_fill_zeros() be run before calculating the measures?
dynamic_range	Dynamic range of the device, in gravity units
calculate_mims	Should MIMS units be calculated?
calculate_ac	Should Activity Counts from the agcounts package be calculated?
flag_data	Should the downstream overlay flag_qc() be run? It will be executed after fix_zeros before any measure calculation
flags	the flags to calculate, passed to the downstream overlay flag_qc()
ensure_all_time	if TRUE, then all times from the first to last times will be in the output, even if data during that time was not in the input
verbose	print diagnostic messages
sample_rate	sample rate of data, if not specified in header of object
...	additional arguments to pass to MIMSunit::mims_unit()
allow_truncation	truncate small values

Value

A data set with the calculated features

Examples

```
file = actiread::acti_example_gt3x()
res = actiread::acti_read_gt3x(file, verbose = FALSE)
res = res[1:12000, ]
measures = acti_calculate_measures(
  res,
  dynamic_range = NULL,
  calculate_mims = FALSE,
  calculate_ac = FALSE,
  flag_data = FALSE
)
auc = acti_calculate_auc(res)

mims = acti_calculate_mims(res, dynamic_range = NULL)

if (requireNamespace("data.table", quietly = TRUE)) {
   dt = data.table::as.data.table(res)
   out = acti_calculate_measures(dt, calculate_mims = FALSE, flag_data = FALSE,
     calculate_ac = FALSE)
}
```

`acti_calculate_sdt` *Calculate Step Counts via Oak/Forest*

Description

Calculate Step Counts via Oak/Forest

Usage

```
acti_calculate_sdt(data, sample_rate = NULL, ...)
```

Arguments

`data`	A `data.frame`, `AccData` object, or GT3X file with `X`, `Y`, `Z`, and `time`
`sample_rate`	Sample rate in Hz. If omitted, it is taken from the input object when available.
`...`	Additional arguments passed to `walking::sdt_count_steps()`

Value

A tibble with minute-level `time`, `steps` columns.

Examples

```
data = actiread::acti_read_gt3x(actiread::acti_example_gt3x())
steps = acti_calculate_sdt(data)
```

acti_calculate_stepcount

Calculate Step Counts via stepcount

Description

Use the stepcount package to estimate steps from raw accelerometer data and summarize them to minute-level epochs (as opposed to 10s default)

Usage

```
acti_calculate_stepcount(data, sample_rate = NULL, ..., epoch = "1 minute")
```

Arguments

data	A data.frame, AccData object, or GT3X file with X, Y, Z, and time
sample_rate	Sample rate in Hz. If omitted, it is taken from the input object when available.
...	Additional arguments passed to stepcount::stepcount()
epoch	epoch unit to aggregate the data to, passed to lubridate::floor_date(), original output is at 10-seconds

Value

A tibble with minute-level time, steps, and walking columns.

Examples

```
# reticulate::py_require("stepcount==3.11.0", python_version = "3.10", action = "add")
# sc = try({ reticulate::import("stepcount") })
if (stepcount::have_stepcount()) {
  data = actiread::acti_read_gt3x(actiread::acti_example_gt3x())
  steps = acti_calculate_stepcount(data, sample_rate = 100)
  steps = acti_calculate_stepcount(data, model_type = "rf")
}
```

acti_calculate_verisense

Calculate Step Counts via Verisense

Description

Calculate Step Counts via Verisense

Usage

```
acti_calculate_verisense(
  data,
  resample_to_15hz = TRUE,
  method = c("original", "revised"),
  ...
)
```

Arguments

data	A data.frame, AccData object, or GT3X file with X, Y, Z, and time
resample_to_15hz	resample data to 15Hz, passed to walking::estimate_steps_verisense()
method	parameters to estimate walking, either original or revised, passed to walking::estimate_steps_verise
...	Additional arguments passed to walking::estimate_steps_verisense()

Value

A tibble with minute-level time, steps columns.

Examples

```
data = actiread::acti_read_gt3x(actiread::acti_example_gt3x())
steps = acti_calculate_verisense(data)
```

---

acti_calibrate *Calibrate Accelerometer Data using* agcounts

Description

This uses the van Hees calibration method typically exposed through GGIR, implemented here via agcounts::agcalibrate().

Usage

```
acti_calibrate(
  data,
  verbose = TRUE,
  fill_zeroes = TRUE,
  round_after_calibration = TRUE,
  ...
)
```

Arguments

data	Either a GT3X file, AccData object, or data.frame with X/Y/Z and time
verbose	print diagnostic messages, higher number result in higher verbosity
fill_zeroes	Should actibase::acti_fill_zeros() be run before calculating the measures? This trims zero values from the beginning and the end of the time course using last observation carried forward behavior.
round_after_calibration	Should the data be rounded after calibration? Will round to 3 digits
...	Additional arguments to pass to agcounts::agcalibrate()

Value

A calibrated dataset with the same columns as the input data, but with the X/Y/Z values calibrated using the van Hees method.

Examples

```
res = acti_calibrate(data = actiread::acti_example_gt3x())
```

---

acti_count_data *Example Actigraphy/Activity Count Data*

Description

Example Actigraphy/Activity Count Data

Usage

```
acti_count_data
```

Format

A data.frame with the columns

time time at the minute level

axis1 axis1 (Y) counts

axis2 axis2 (X) counts

axis3 axis3 (Z) counts

counts vector magnitude of all 3 axes column

This data was taken from running agcounts::calculate_counts via acti_calculate_counts on actibase::acti_raw_data.

acti_process *Process Count Data*

Description

Process Count Data

Usage

```
acti_process(
  data,
  lfe_select = FALSE,
  method = c("choi", "troiano"),
  use_magnitude = TRUE,
  verbose = TRUE,
  ...
)
```

Arguments

data	A data.frame from acti_calculate_counts that has columns axis1-3 and counts
lfe_select	Apply the Actigraph Low Frequency Extension filter. See agcounts::calculate_counts() higher values are higher levels of verbosity.
method	Method for detecting non-wear, either "choi" or "troiano", corresponding to actigraph.sleepr::apply_choi() or actigraph.sleepr::apply_troiano()
use_magnitude	If TRUE, the magnitude of the vector (axis1, axis2, axis3) is used to measure activity; otherwise the axis1 value is used.
verbose	print diagnostic messages. Either logical or integer, where
...	additional arguments to pass to actigraph.sleepr function

Value

A data frame containing activity counts and wear-time indicators.

Note

For acti_process_gt3x, the ... argument are passed to actiread::acti_read_gt3x()

mims_default_extrapolation
Default MIMS worker functions

Description

Default MIMS worker functions

Usage

```
mims_default_extrapolation(data, dynamic_range = NULL)

mims_default_interpolation(data)

mims_default_filtering(data)
```

Arguments

data	data set of data, usually time and X/Y/Z. Usually from `actiread::acti_read_gt3x()`
dynamic_range	dynamic range of the data. Will be passed to `actibase::get_dynamic_range()`

Value

A data set of data

mims_default_processing
Default MIMS Pre-processing

Description

Default MIMS Pre-processing

Usage

```
mims_default_processing(
  data,
  use_extrapolation = TRUE,
  use_filtering = TRUE,
  verbose = TRUE,
  dynamic_range = NULL,
  round_after_processing = FALSE
)
```

Arguments

data	Data set of raw accelerometry values, usually time and X/Y/Z. Usually from `actiread::acti_read_gt3x()`
use_extrapolation	If TRUE the function will apply extrapolation algorithm to the input signal, otherwise it will skip extrapolation but only linearly interpolate the signal to 100Hz.
use_filtering	If TRUE the function will apply bandpass filtering to the input signal, otherwise it will skip the filtering.
verbose	print diagnostic messages
dynamic_range	the dynamic ranges of the input signal. Passed to `actimetrics::mims_default_extrapolation()`. Only needed if `use_extrapolation = TRUE`
round_after_processing	Should the result be rounded to 3 decimal values after processing, to make similar to standard accelerometry?

Value

A processed data set

---

py_acti_calculate_forest

Perform step count calculation in a separate Python environment

---

Description

Perform step count calculation in a separate Python environment

Usage

```
py_acti_calculate_forest(
  ...,
  pyenv_function = function() {
     reticulate::import("forest")
 },
  show = FALSE
)
```

Arguments

...	arguments passed to `acti_calculate_forest()`
pyenv_function	function that loads the `forest` Python package. By default, it uses `reticulate::py_import("forest")` to import the package. If this function has an `args` argument, the output of `pyenv_function` will be re-assigned to `args`.
show	Logical, whether to show the standard output on the screen while the child process is running, passed to `callr::r()`

Value

The output from acti_calculate_forest().

Examples

```
Sys.setenv("SSQ_PARALLEL" = 0)
data = actiread::acti_read_gt3x(actiread::acti_example_gt3x())
steps = py_acti_calculate_forest(data, sample_rate = 100)
```

---

py_acti_calculate_stepcount

Perform step count calculation in a separate Python environment

---

Description

Perform step count calculation in a separate Python environment

Usage

```
py_acti_calculate_stepcount(
  ...,
  pyenv_function = function() {
     stepcount::py_require_stepcount()
 },
  show = FALSE
)
```

Arguments

...	arguments passed to acti_calculate_stepcount()
pyenv_function	function that loads the stepcount Python package. By default, it uses reticulate::py_import("stepco to import the package. If this function has an args argument, the output of pyenv_function will be re-assigned to args.
show	Logical, whether to show the standard output on the screen while the child process is running, passed to callr::r()

Value

The output from acti_calculate_stepcount(). A tibble with minute-level time, steps, and walking columns.

Examples

```
if (stepcount::have_stepcount()) {
  data = actiread::acti_read_gt3x(actiread::acti_example_gt3x())
  steps = py_acti_calculate_stepcount(data, sample_rate = 100)
}
```

Index