

Package: actimetrics (via r-universe)

October 20, 2026

Type Package

Title Create Metrics Actigraphy and Activity Analysis

Version 0.9.0

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](https://doi.org/10.1139/apnm-2013-0173)> and Sadeh (1994) <[doi:10.1093/sleep/17.3.201](https://doi.org/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), agcounts (>= 0.6.7), callr

Encoding UTF-8

VignetteBuilder knitr

Imports actibase, actiread, actigraph.sleepr (>= 0.3.1), dplyr, lubridate, assertthat, purrr, readr, 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 <muschelli2@gmail.com>

Repository <https://cran.r-universe.dev>

Date/Publication 2026-10-20 17:20:49 UTC
RemoteUrl <https://github.com/cran/actimetrics>
RemoteRef HEAD
RemoteSha a38c5b1845896f751a590bd88a66c5ad3dfd4347

Contents

acti_apply_tudor_locke	2
acti_apply_tudor_locke_diary	3
acti_calculate_ai	4
acti_calculate_counts	5
acti_calculate_forest	7
acti_calculate_measures	8
acti_calculate_sdt	10
acti_calculate_stepcount	11
acti_calculate_verisense	11
acti_calibrate	12
acti_count_data	13
acti_process	14
acti_process_study	15
mims_default_extrapolation	16
mims_default_processing	16
py_acti_calculate_forest	17
py_acti_calculate_stepcount	18
Index	20

acti_apply_tudor_locke
Apply Tudor–Locke sleep period detection

Description

Delegates to `actigraph.sleepr::apply_tudor_locke()`. Input must be 60-second epoch data with activity counts and a sleep state. Data from `acti_apply_cole_kripke()` is accepted directly. Data labeled by `acti_apply_tudor_locke_diary()` is also accepted: its logical sleep column is converted to the S/W states expected downstream.

Usage

`acti_apply_tudor_locke(data, ..., sleep_col = "sleep")`

Arguments

<code>data</code>	Epoch-level data containing activity counts and sleep state.
<code>...</code>	Arguments passed to <code>actigraph.sleepr::apply_tudor_locke()</code> .
<code>sleep_col</code>	Name of the sleep-state column in data, either logical (as from acti_apply_tudor_locke_diary()) or S/W. It is used as the sleep column expected by <code>actigraph.sleepr</code> .

Value

A data frame summarizing detected sleep periods.

`acti_apply_tudor_locke_diary`

Apply a sleep diary to epoch data for Tudor-Locke

Description

Adds a logical sleep column to epoch data from diary onset and wakeup times. The result can be passed to [acti_apply_tudor_locke\(\)](#) when it also has activity counts and represents 60-second epochs. The wrapper converts the logical diary labels to the S/W states required downstream.

Usage

```
acti_apply_tudor_locke_diary(
  data,
  sleep_diary,
  time_col = "time",
  diary_onset_col = "onset",
  diary_wakeup_col = "wakeup",
  sleep_col = "sleep"
)
```

Arguments

<code>data</code>	Epoch-level data containing a POSIXct timestamp column.
<code>sleep_diary</code>	A data frame with POSIXct sleep onset and wakeup columns. Each interval may span midnight; multiple days are supported.
<code>time_col</code>	Name of the POSIXct timestamp column in data.
<code>diary_onset_col</code>	Name of the diary onset column.
<code>diary_wakeup_col</code>	Name of the diary wakeup column.
<code>sleep_col</code>	Name of the logical sleep column to add or replace.

Value

data with a logical `sleep_col`, TRUE from onset (inclusive) to wakeup (exclusive).

acti_calculate_ai	<i>Calculate the Activity Index (AI)</i>
-------------------	--

Description

Computes the Activity Index of Bai et al. (2016) from raw tri-axial accelerometry. For each 1-second window t , with $\sigma_m^2(t)$ the variance of axis $m \in \{X, Y, Z\}$ over the samples in that second,

$$AI(t) = \sqrt{\frac{1}{3} \sum_m \sigma_m^2(t)}.$$

The per-second values are then summed within each `unit` (e.g. a minute).

Usage

```
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
)
```

Arguments

<code>data</code>	An object with columns X, Y, and Z and a time column, or an object of class <code>AccData</code>
<code>unit</code>	length of time to sum the per-second AI over, passed to <code>lubridate::floor_date()</code>
<code>ensure_all_time</code>	if TRUE, all times from the first to the last <code>unit</code> are in the output, even those with no input data
<code>verbose</code>	print diagnostic messages

Details

This is the unadjusted index (sometimes written AI_0): it does not subtract or normalise by the device's systematic noise variance $\bar{\sigma}_m^2$, which the original definition estimates from a stationary period. It differs from `AI_DEFINED` returned by `acti_calculate_mad()`, which takes the variances over the whole `unit` rather than summing 1-second values.

A 1-second variance needs at least two samples, so seconds with fewer than two samples are NA and are skipped when summing; a `unit` with no valid seconds is NA. Data sampled at under 2 Hz therefore give all-NA AI (with a warning).

Value

A data.frame (or data.table if data is one) with columns HEADER_TIME_STAMP and AI.

References

Bai J, Di C, Xiao L, et al. (2016). An Activity Index for Raw Accelerometry Data and Its Comparison with Other Activity Metrics. PLoS ONE 11(8): e0160644. doi:[10.1371/journal.pone.0160644](https://doi.org/10.1371/journal.pone.0160644)

Examples

```
file = actiread::acti_example_gt3x()
res = actiread::acti_read_gt3x(file, verbose = FALSE)
ai = acti_calculate_ai(res[1:12000, ])
```

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,
  fix_first_second = TRUE,
  round_digits = NULL
)

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_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
verbose	print diagnostic messages. Either logical or integer, where higher values are higher levels of verbosity.
fix_first_second	Fix the first second bug in agcounts. Appends replicated data of the first record if the first record is not an "even" epoch (e.g. 60 second epoch and data does not start at a 00 second).
round_digits	If not NULL, the number of decimal places the X/Y/Z values are rounded to after resampling and before counting. Use 3 to match ActiLife, which rounds GT3X data to 3 decimals (half away from zero, so 0.0625 becomes 0.063, unlike base::round()). Useful if the data were read with more digits. Default NULL does not round.
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

Details

The sampling frequency is only needed when `resample = FALSE`; it is taken from the `sample_rate` attribute (or estimated from time).

`acti_calculate_wear()`, `acti_apply_cole_kripke()`, and `acti_apply_sadeh()` require 60-second epochs. Cole-Kripke, Sadeh, and `acti_calculate_wear(use_magnitude = FALSE)` score the vertical axis, so data must contain axis1; vector-magnitude-only counts are an error.

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 <code>data.frame</code> , <code>AccData</code> 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_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

<code>data</code>	An object with columns X, Y, and Z or an object of class <code>AccData</code>
<code>unit</code>	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 <code>lubridate::period()</code> constructor are allowed.
<code>fix_zeros</code>	Should <code>actibase::acti_fill_zeros()</code> be run before calculating the measures?
<code>dynamic_range</code>	Dynamic range of the device, in gravity units
<code>calculate_mims</code>	Should MIMS units be calculated?
<code>calculate_ac</code>	Should Activity Counts from the <code>agcounts</code> package be calculated?
<code>flag_data</code>	Should the downstream overlay <code>flag_qc()</code> be run? It will be executed after <code>fix_zeros</code> before any measure calculation
<code>flags</code>	the flags to calculate, passed to the downstream overlay <code>flag_qc()</code>
<code>ensure_all_time</code>	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
<code>verbose</code>	print diagnostic messages
<code>sample_rate</code>	sample rate of data, if not specified in header of object
<code>...</code>	additional arguments to pass to <code>MIMSunit::mims_unit()</code>
<code>allow_truncation</code>	truncate small values to 0 and mark epochs with impossibly large values as invalid (-0.01 for all axes and the total), following <code>MIMSunit</code>

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	<i>Calculate Step Counts via SDT</i>
--------------------	--------------------------------------

Description

Calculate Step Counts via SDT

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_verisense()
...	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	<i>Calibrate Accelerometer Data using agcounts</i>
----------------	--

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

<code>data</code>	Either a GT3X file, <code>AccData</code> object, or <code>data.frame</code> with X/Y/Z and time
<code>verbose</code>	print diagnostic messages, higher number result in higher verbosity
<code>fill_zeroes</code>	Should <code>actibase::acti_fill_zeros()</code> 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.
<code>round_after_calibration</code>	Should the data be rounded after calibration? Will round to 3 digits
<code>...</code>	Additional arguments to pass to <code>agcounts::agcalibrate()</code>

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())
```

<code>acti_count_data</code>	<i>Example Actigraphy/Activity Count Data</i>
------------------------------	---

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	<i>Process Raw data to Counts</i>
--------------	-----------------------------------

Description

Process Raw data to Counts

Usage

```
acti_process(
  data,
  lfe_select = FALSE,
  method = c("choi", "troiano"),
  use_magnitude = TRUE,
  calibrate = FALSE,
  verbose = TRUE,
  ...,
  min_required = 1368L,
  round_digits = NULL
)
```

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
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.
calibrate	Logical. If TRUE, the data will be calibrated using acti_calibrate()
verbose	print diagnostic messages. Either logical or integer, where higher values are higher levels of verbosity.
...	additional arguments to pass to actigraph.sleepr function
min_required	Number of minutes required in a day to be called included. No day inclusion is run if NULL.
round_digits	Number of decimal places X/Y/Z are rounded to (half away from zero) after re-sampling to 30Hz and before counting. The default NULL uses 3 for GT3X data (read with actiread::acti_read_gt3x()) and does not round other data. Use NA to never round.

Details

To match ActiLife for GT3X files, a GT3X path is read with digits = 8 (unless digits is given in ...), resampled, and then rounded to 3 digits. ActiLife rounds the scaled samples to 3 digits, half away from zero. Reading at 8 digits and then rounding gives exactly those values, but rounding values already read at 5 digits does not (e.g. 0.1234996 -> 0.12350 -> 0.124).

Value

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

Note

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

<code>acti_process_study</code>	<i>Process all supported actigraphy files in a directory</i>
---------------------------------	--

Description

Files are read according to their extension, passed to `acti_process()`, and saved as individual RDS files in `outdir`. Supported extensions are GT3X, CWA, GeneActiv BIN, and CSV, including `.gz` and `.xz` compressed suffixes for the supported formats described below.

Usage

```
acti_process_study(directory, outdir, verbose = TRUE, ...)
```

Arguments

<code>directory</code>	Directory containing raw actigraphy files.
<code>outdir</code>	Directory in which processed RDS files are saved. Created if needed.
<code>verbose</code>	Logical; report progress for each file. Failures are always reported with a warning.
<code>...</code>	Arguments passed to <code>acti_process()</code> .

Value

Invisibly, a character vector of paths to the saved RDS files. Files that failed are listed in its `failed` attribute: a named character vector of error messages, named by input file. Each RDS contains the processed data, with the raw reader header attached as the header attribute when available.

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 <code>actiread::acti_read_gt3x()</code>
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 <code>actimetrics::mims_default_extrapolation()</code> . Only needed if <code>use_extrapolation = TRUE</code>
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 <code>reticulate::py_import("forest")</code> to import the package. If this function has an <code>args</code> argument, the output of <code>pyenv_function</code> will be re-assigned to <code>args</code> .
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)
if (reticulate::py_module_available("forest")) {
  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

<code>...</code>	arguments passed to <code>acti_calculate_stepcount()</code>
<code>pyenv_function</code>	function that loads the stepcount Python package. By default, it uses <code>stepcount::py_require_stepcount()</code> to set up the package. If this function has an <code>args</code> argument, the output of <code>pyenv_function</code> will be re-assigned to <code>args</code> .
<code>show</code>	Logical, whether to show the standard output on the screen while the child process is running, passed to <code>callr::r()</code>

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

* datasets

- acti_count_data, 13
- acti_apply_cole_kripke
 - (acti_calculate_counts), 5
- acti_apply_cole_kripke(), 2
- acti_apply_sadeh
 - (acti_calculate_counts), 5
- acti_apply_tudor_locke, 2
- acti_apply_tudor_locke(), 3
- acti_apply_tudor_locke_diary, 3
- acti_apply_tudor_locke_diary(), 2, 3
- acti_calculate_activity_index
 - (acti_calculate_ai), 4
- acti_calculate_ai, 4
- acti_calculate_ai_defined
 - (acti_calculate_measures), 8
- acti_calculate_auc
 - (acti_calculate_measures), 8
- acti_calculate_counts, 5, 13
- acti_calculate_enmo
 - (acti_calculate_measures), 8
- acti_calculate_fast_mims
 - (acti_calculate_measures), 8
- acti_calculate_flags
 - (acti_calculate_measures), 8
- acti_calculate_forest, 7
- acti_calculate_forest(), 17, 18
- acti_calculate_mad
 - (acti_calculate_measures), 8
- acti_calculate_mad(), 4
- acti_calculate_measures, 8
- acti_calculate_mims
 - (acti_calculate_measures), 8
- acti_calculate_n_idle
 - (acti_calculate_measures), 8
- acti_calculate_nonwear
 - (acti_calculate_counts), 5
- acti_calculate_sdt, 10
- acti_calculate_stepcount, 11
- acti_calculate_stepcount(), 18
- acti_calculate_verisense, 11
- acti_calculate_wear
 - (acti_calculate_counts), 5
- acti_calibrate, 12
- acti_calibrate(), 14
- acti_count_data, 13
- acti_process, 14
- acti_process(), 15
- acti_process_study, 15
- actibase::acti_raw_data, 13
- actibase::acti_resample, 6
- actiread::acti_read_gt3x(), 14, 15
- agcounts::calculate_counts, 6, 13, 14
- base::round(), 6
- callr::r(), 17, 18
- lubridate::floor_date(), 4, 11
- mims_default_extrapolation, 16
- mims_default_filtering
 - (mims_default_extrapolation), 16
- mims_default_interpolation
 - (mims_default_extrapolation), 16
- mims_default_processing, 16
- py_acti_calculate_forest, 17
- py_acti_calculate_stepcount, 18
- stepcount::stepcount(), 11
- walking::estimate_steps_verisense(), 12
- walking::find_walking(), 7
- walking::sdt_count_steps(), 10