Package 'stepmetrics'

September 16, 2025

Type Package

Title Calculate Step and Cadence Metrics from Wearable Data

Version 1.0.1

Description Provides functions to calculate step- and cadence-based metrics from timestamped accelerometer and wearable device data. Supports CSV and AGD files from 'ActiGraph' devices, CSV files from 'Fitbit' devices, and step counts derived with R package 'GGIR' <https://github.com/wadpac/GGIR>, with automatic handling of epoch lengths from 1 to 60 seconds. Metrics include total steps, cadence peaks, minutes and steps in predefined cadence bands, and time and steps in moderate-to-vigorous physical activity (MVPA). Methods and thresholds are informed by the literature, e.g., Tudor-Locke and Rowe (2012) <doi:10.2165/11599170-000000000-00000>, Barreira et al. (2012) <doi:10.1249/MSS.0b013e318254f2a3>, and Tudor-Locke et al. (2018) <doi:10.1136/bjsports-2017-097628>. The package record is also available on Zenodo (2023) <doi:10.5281/zenodo.7858094>.

License AGPL (>= 3)

Depends R (>= 3.5.0)

Encoding UTF-8

RoxygenNote 7.3.2

Imports PhysicalActivity, tools, stats, utils

Suggests testthat (>= 3.0.0), RSQLite, spelling

Config/testthat/edition 3

URL https://github.com/jhmigueles/stepmetrics

BugReports https://github.com/jhmigueles/stepmetrics/issues

Language en-US

NeedsCompilation no

Author Jairo H Migueles [aut, cre],
Elroy J Aguiar [fnd] (Funding and data support),
University of Alabama [fnd]

Maintainer Jairo H Migueles <jairo@jhmigueles.com>

Repository CRAN

Date/Publication 2025-09-16 06:40:02 UTC

Contents

`define_day_indices` *Generate sequential day indices from ISO 8601 timestamps*

Description

Converts a vector of ISO 8601 timestamps into sequential day indices (1, 2, 3, ...), where each unique calendar date corresponds to a unique integer. This is useful for looping over or summarizing data by day when working with minute-level time series from wearables.

Usage

```
define_day_indices(ts)
```

Arguments

`ts`	Character vector of ISO 8601 timestamps (e.g., '"2024-06-26T23:45:00+0100"'). Time zone offsets are handled correctly when converting to 'Date'.

Value

An integer vector of the same length as 'ts', where each element gives the index of the day the timestamp belongs to. The first unique date encountered is assigned index '1', the second '2', and so on.

See Also

[as.POSIXct()], [as.Date()]

Examples

```
ts <- c("2024-06-26T23:45:00+0100",
        "2024-06-27T00:15:00+0100",
        "2024-06-27T14:30:00+0100")
define_day_indices(ts)
# Returns: c(1, 2, 2)
```

`get_cadence_bands` *Calculate minutes and steps spent in cadence bands*

Description

Splits a minute-based vector of cadence values (steps per minute) into predefined bands and reports both:

- the number of minutes spent in each band
- the number of steps accumulated in each band

Usage

```
get_cadence_bands(x, bands = c(0, 1, 20, 40, 60, 80, 100, 120, Inf))
```

Arguments

`x`	Numeric vector of cadence values (steps per minute), where each element represents one minute of the day.
`bands`	Numeric vector of break points that define the cadence bands. Defaults to `c(0, 1, 20, 40, 60, 80, 100, 120, Inf)`, which produces the bands 0, 1–19, 20–39, 40–59, 60–79, 80–99, 100–119, and $\geq$120 spm.

Value

A list with three elements:

minutes Named numeric vector with minutes spent in each band.

steps Named numeric vector with steps accumulated in each band.

names Character vector of variable names in the format `"CAD_band_<lower>_<upper>_spm"`.

Examples

```
# Simulate 1 day of cadence values (1440 minutes)
set.seed(123)
cad <- sample(0:150, size = 1440, replace = TRUE)

out <- get_cadence_bands(cad)
out$minutes  # minutes in each band
out$steps    # steps in each band
```

`get_cadence_peaks` *Calculate cadence peak metrics*

Description

Cadence peaks represent the mean steps-per-minute (spm) achieved during the highest-activity minutes of a day. For example, the 30-minute cadence peak is the average cadence across the 30 highest cadence minutes of that day, regardless of whether they occur consecutively.

This function calculates cadence peaks for user-specified intervals and also reports how many of the minutes within each interval contain zero steps (useful for quality checks).

Usage

```
get_cadence_peaks(x, peaks = c(1, 30, 60))
```

Arguments

`x`	Numeric vector of cadence values (steps per minute) for the day of interest. Each element should represent one minute.
`peaks`	Numeric vector of integers specifying which cadence peaks to compute. For example, ‘c(1, 30, 60)‘ (default) produces the 1-minute, 30-minute, and 60-minute cadence peaks.

Details

Cadence peaks are calculated by:

1. Sorting all minute-level cadence values in descending order.
2. Selecting the top *n* minutes, where *n* = peak length.
3. Averaging those values to compute the cadence peak.
4. Counting how many of those top *n* minutes contain zero steps.

Value

A list with two elements:

values Numeric vector with cadence peak values (spm) followed by the corresponding counts of zero-minute values within each peak window.

names Character vector of variable names corresponding to the ‘values‘, in the format:

- ‘"CAD_pk<peak>_spm"‘ for the cadence peak value
- ‘"CAD_nZeroes_pk<peak>"‘ for the number of zero minutes

See Also

[get_cadence_bands()]

Examples

```
# Simulate one day of cadence values (1440 minutes)
set.seed(123)
cad <- sample(0:150, size = 1440, replace = TRUE)

# Calculate 1-min, 30-min, and 60-min cadence peaks
get_cadence_peaks(cad, peaks = c(1, 30, 60))
```

`isGGIRoutput` *Check if a directory is a valid GGIR output folder for stepmetrics*

Description

Determines whether a given directory corresponds to a valid GGIR output directory that can be used with **stepmetrics**. Several conditions are checked in sequence:

1. Path exists and is a directory.
2. Directory name begins with ‘"output_"‘.
3. Contains a ‘meta/‘ subfolder.
4. Contains a ‘meta/ms2.out/‘ subfolder (GGIR part 2 milestone data).
5. Contains at least one ‘*.RData‘ file in ‘ms2.out‘.
6. The loaded object `IMP$metashort` includes a step column (with ‘"step"‘ in its name).

If any of these checks fail, the function returns `FALSE` and issues a warning describing the missing requirement.

Usage

```
isGGIRoutput(path)
```

Arguments

`path`	Character. Path to the directory to be tested.

Value

Logical scalar. Returns `TRUE` if the directory appears to be valid GGIR output suitable for stepmetrics, otherwise `FALSE`.

Note

- A warning is issued if the directory looks like GGIR output but is missing required parts (e.g., part 2 milestone data or step counts in `IMP$metashort`). - This function loads the first available RData file in `meta/ms2.out/` to verify the presence of step counts.

See Also

[step.metrics()]

Examples

```
## Not run:
# GGIR output folder
ggir_output_dir = system.file("extdata", "testfiles_GGIR", "output_test", package = "stepmetrics")
isGGIRoutput(ggir_output_dir)

# Non-GGIR directory
isGGIRoutput("C:/mystudy/rawdata/")

## End(Not run)
```

`readFile` *Read and standardize minute-level step data for one participant*

Description

Reads one or more files for a single participant and returns a clean, minute-level data frame with two columns: ‘timestamp‘ and ‘steps‘. The function auto-detects common file formats and timestamp layouts, fixes ActiGraph CSV headers/metadata when present, and aggregates to a 60-second epoch if input data are recorded at sub-minute resolution.

Supported input formats

- **CSV**: generic CSVs and ActiGraph exports (header lines and delimiters auto-detected; handles date/time split columns).
- **AGD**: ActiGraph binary files via **PhysicalActivity**.
- **RData**: GGIR output (`IMP$metashort`).

Usage

```
readFile(path, time_format = c())
```

Arguments

`path`	Character vector. Path(s) to the file(s) containing timestamp and step data for one participant. When multiple files are provided, they are concatenated in the order given.
`time_format`	Character (optional). Explicit timestamp format string (as used by `strptime`) to override auto-detection for CSV inputs. If `NULL`, common formats are tried automatically.

Details

- **CSV handling:** The function detects and skips ActiGraph header lines (typically 10), infers the field separator (comma/semicolon), and reconstructs a single timestamp when date and time are stored in separate columns. If no explicit timestamp column exists (rare ActiGraph cases), it reconstructs one from the file metadata (start time + epoch).
- **AGD handling:** AGD files are read with `readActigraph`; the recording start and epoch length are obtained from the embedded database and used to build a regular timestamp sequence.
- **Step column detection:** The column containing step counts is inferred by matching names containing *"step"* or *"value"*; if multiple candidates are present, the column with higher variability is chosen.
- **Epoch standardization:** If the input epoch is shorter than 60 seconds, rows are aggregated by summing steps to 1-minute bins. Epochs longer than 60 seconds are currently unsupported.

Value

A `data.frame` with two columns:

`timestamp` Character vector of ISO-8601 datetimes (`"YYYY-MM-DDTHH:MM:SS(+/-)ZZZZ"`) for CSV/AGD inputs. For GGIR `RData` inputs, timestamps are carried through as present in `IMP$metashort`.

`steps` Numeric vector of steps per minute. If the source data have sub-minute epochs, values are summed to 60-second bins. Epochs longer than 60 seconds are not supported and trigger an error.

Time zones

For CSV/AGD inputs, timestamps are returned in ISO-8601 with an explicit offset. If `time_format` is provided, it is passed directly to `strptime` for parsing; otherwise a set of common formats is attempted.

See Also

`step.metrics`, `get_cadence_bands`

Examples

```
# Fitbit csv
fitbit_csv = system.file("extdata", "testfiles_fitbit",
                         "S001_d1_1min_epoch.csv", package = "stepmetrics")
df <- readFile(fitbit_csv)

# ActiGraph AGD
actigraph_agd = system.file("extdata", "testfiles_agd", "3h30sec.agd", package = "stepmetrics")
df <- readFile(actigraph_agd)
```

`step.metrics` *Calculate and export daily and person-level step and cadence metrics*

Description

This function processes epoch-level step count files (raw exports or GGIR output) and derives a comprehensive set of **daily** and **person-level** metrics. Metrics include total steps, cadence peaks, time and steps accumulated in predefined cadence bands, and time and steps in moderate, vigorous, and moderate-to-vigorous physical activity (MPA, VPA, MVPA).

The function writes two types of summary CSVs:

- **Day-level:** One file per participant per day, stored in 'outputdir/daySummary/'.
- **Person-level:** A single file with aggregated averages across valid days per participant, stored as 'outputdir/personSummary.csv'.

Usage

```
step.metrics(
  datadir,
  outputdir = "./",
  idloc = "_",
  cadence_bands = c(0, 1, 20, 40, 60, 80, 100, 120, Inf),
  cadence_peaks = c(1, 30, 60),
  cadence_MOD = 100,
  cadence_VIG = 130,
  includedaycrit = 10,
  includeawakecrit = NULL,
  includedaylengthcrit = 23,
  exclude_pk30_0 = TRUE,
  exclude_pk60_0 = TRUE,
  time_format = NULL,
  verbose = TRUE
)
```

Arguments

`datadir`	Character. Path to the directory containing the step data. If processing GGIR output, provide the GGIR output folder (its name starts with '"output_"'); the function will then look inside 'meta/ms2.out/'.
`outputdir`	Character. Directory where results should be stored. Subfolders will be created as needed ('daySummary/').
`idloc`	Character (default = '"_"'). Delimiter used to extract participant IDs from filenames (ID is expected before this string).
`cadence_bands`	Numeric vector (default = 'c(0, 1, 20, 40, 60, 80, 100, 120, Inf)'). Breakpoints (in steps/min) used to compute time and steps per cadence band.

cadence_peaks	Numeric vector (default = 'c(1, 30, 60)'). Cadence peak values (e.g., peak 30 = mean of the top 30 cadence minutes).
cadence_MOD	Numeric (default = 100). Threshold cadence (steps/min) for moderate physical activity.
cadence_VIG	Numeric (default = 130). Threshold cadence (steps/min) for vigorous physical activity.
includedaycrit	Numeric (default = 10). Minimum wear time in hours for a day to be considered valid.
includeawakecrit	Numeric (default = NULL). If GGIR part 5 outputs are available, use the proportion of awake time instead of wear time to define valid days.
includedaylengthcrit	Numeric (default = 23). Minimum day length (hours) for a day to be valid (only relevant if using GGIR part 5 outputs).
exclude_pk30_0	Logical (default = TRUE).
exclude_pk60_0	Logical (default = TRUE). Exclude days with zero values in the 60-minute cadence peak.
time_format	Character (default = NULL). Time format used when reading non-GGIR step files.
verbose	Logical (default = TRUE). Whether to print progress messages.

Details

For each participant and day, the function computes:

- Total steps per day
- Cadence peak metrics (e.g., peak 1, 30, 60 minutes)
- Minutes and steps in each cadence band
- Minutes in MPA, VPA, and MVPA
- Steps accumulated in MPA, VPA, and MVPA
- Recording duration, valid wear time, and awake time (if GGIR available)

Person-level outputs include plain, weighted (weekday/weekend), and stratified (weekday/weekend separately) averages of all variables.

Value

This function does not return an object. It writes:

- <ID>_DaySum.csv in 'outputdir/daySummary/' with daily metrics.
- personSummary.csv in 'outputdir/' with person-level averages.

Examples

```
datadir = system.file("extdata", "testfiles_fitbit", package = "stepmetrics")
step.metrics(datadir = datadir, outputdir = tempdir())
```

Index