Package: rabbit (via r-universe)

November 13, 2026

Title Rabbitly Process Data From Triaxial Accelerometers

Version 0.0.0.9000

Description Rabbitly Process Data From Triaxial Accelerometers.

License MIT + file LICENSE

Encoding UTF-8

Depends R (>= 4.0.0)

Imports dplyr, arrow, data.table, RcppRoll, lubridate

Suggests covr, purrr, ggplot2, knitr, rmarkdown, testthat (>= 3.0.0)

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

Config/testthat/edition 3

Config/pak/sysreqs cmake libxml2-dev libssl-dev

Repository https://traitecoevo.r-universe.dev

Date/Publication 2026-08-20 00:45:36 UTC

RemoteUrl https://github.com/traitecoevo/rabbit

RemoteRef HEAD

RemoteSha 1705756640f6e3c627cbadcbb06b54fb696b9537

Contents

benchmark_movement_dynamics 2
crop_ends 2
extract_movement_dynamics 3
generate_fake_data 4
standardise_data 5

Index **7**

`benchmark_movement_dynamics` *Benchmark rolling methods used by movement dynamics*

Description

Compare elapsed runtime for fast (RcppRoll) and base (pure R) methods for both `roll_sum()` and `extract_movement_dynamics()`.

Usage

```
benchmark_movement_dynamics(df, window_size = 50, iterations = 3)
```

Arguments

`df`	A data frame containing columns `time`, `x`, `y`, `z`.
`window_size`	An integer specifying the size of the rolling window.
`iterations`	Number of repeated timings per task/method.

Value

A tibble with columns `task`, `method`, `iteration`, and `elapsed_sec`.

`crop_ends` *Crop data to specified start and end times, to remove records at the beginning and end of each file. This is useful for removing records collected before the device was attached or after it was removed. crop_diurnal defaults to cropping dataframe to the first instance of 01:00:00 and the last instance of 01:00:00. crop_nocturnal defaults to cropping dataframe to the first instance of 12:00:00 and the last instance of 12:00:00. crop_ends is a more general function that allows users to specify any start and end times. crop_diurnal*

Description

Crop data to specified start and end times, to remove records at the beginning and end of each file. This is useful for removing records collected before the device was attached or after it was removed. crop_diurnal defaults to cropping dataframe to the first instance of 01:00:00 and the last instance of 01:00:00. crop_nocturnal defaults to cropping dataframe to the first instance of 12:00:00 and the last instance of 12:00:00. crop_ends is a more general function that allows users to specify any start and end times. crop_diurnal

Usage

```
crop_ends(df, start_time = "00:00:00", end_time = "24:00:00")

crop_ends_diurnal(df, start_time = "01:00:00", end_time = "01:00:00")

crop_ends_nocturnal(df, start_time = "12:00:00", end_time = "12:00:00")
```

Arguments

df	A dataframe containing the data.
start_time	The desired start time
end_time	The desired end time

`extract_movement_dynamics`

This function takes standardised accelerometer data and caluclates a set of metrics of movement dynamics across a sliding window of specified number of datapoints. Caluclate metrics include including means, variances, covariances, of xyz, as well as overall dynamic body acceleration (ODBA) and vectorial dynamic body acceleration (VDBA). The function is designed to be fast, using the RcppRoll package to calculate rolling means and sums in C++.

Description

This function takes standardised accelerometer data and caluclates a set of metrics of movement dynamics across a sliding window of specified number of datapoints. Caluclate metrics include including means, variances, covariances, of xyz, as well as overall dynamic body acceleration (ODBA) and vectorial dynamic body acceleration (VDBA). The function is designed to be fast, using the RcppRoll package to calculate rolling means and sums in C++.

Usage

```
extract_movement_dynamics(df, window_size = 50, method = c("fast", "base"))
```

Arguments

df	A data frame containing the data.
window_size	An integer specifying the size of the rolling window.
method	A string specifying the method to use for rolling calculations. Options are "fast" (using RcppRoll) or "base" (using pure R implementations). The default is "fast".

Value

A data frame with movement dynamics metrics calculated across the sliding window.

Examples

```
#' @examples
file_in = system.file("extdata", "raw_Pic2Jan_50000.parquet", package = "rabbit")
df <-
  standardize_data(file_in = file_in, vars = c("Timestamp", "accX", "accY", "accZ")) |>
  extract_movement_dynamics()
```

`generate_fake_data` *Generate fake ample accelerometer data*

Description

Creates synthetic triaxial accelerometer data with 1-second timestamps. The `accZ` axis is centered around 1 (gravity-like baseline), while `accX` and `accY` are centered around 0.

Usage

```
generate_fake_data(n)
```

Arguments

`n`	Integer. Number of rows to generate.

Value

A data frame with columns:

timestamp Character timestamp in format `"%d/%m/%Y %H:%M:%S"` (UTC).

accX Simulated x-axis acceleration values.

accY Simulated y-axis acceleration values.

accZ Simulated z-axis acceleration values.

Examples

```
df <- generate_fake_data(1000)
head(df)
```

`standardise_data` *Standardise accelerometer data*

Description

This function accepts either a dataframe or filename (*.csv or *.parquet), checks that the time and x,y,z columns can be properly processed, auto-detects common timestamp formats, and prints out summary statistics about the data.

Usage

```
standardise_data(
  df,
  file_in = NULL,
  vars = c("Timestamp", "X", "Y", "Z"),
  time_function = lubridate::dmy_hms,
  timezone = "Australia/Adelaide",
  ...
)

standardize_data(
  df,
  file_in = NULL,
  vars = c("Timestamp", "X", "Y", "Z"),
  time_function = lubridate::dmy_hms,
  timezone = "Australia/Adelaide",
  ...
)
```

Arguments

`df`	A data frame containing the data
`file_in`	A string representing the path to the input file (CSV or parquet).
`vars`	A character vector of length 4 specifying the names of the columns in df that contain the variables `time`, `X`, `Y`, `Z` from the accelerometer. The default is c("Timestamp","X","Y","Z").
`time_function`	A function to convert the time column to POSIXct. The default is lubridate::dmy_hms, which parses date-times with year, month, and day, hour, minute, and second components.
`timezone`	Timezone of where accelerometer was used
`...`	Other arguments to pass into read in function

Value

A data frame containing the standardised data, with the time column converted to POSIXct type.

Examples

```
file_in = system.file("extdata", "raw_Pic2Jan_50000.parquet", package = "rabbit")
df <-
  standardise_data(file_in = file_in, vars = c("Timestamp", "accX", "accY", "accZ"))
```

Index

benchmark_movement_dynamics, 2

crop_ends, 2
crop_ends_diurnal *(*crop_ends*)*, 2
crop_ends_nocturnal *(*crop_ends*)*, 2

extract_movement_dynamics, 3

generate_fake_data, 4

standardise_data, 5
standardize_data *(*standardise_data*)*, 5