Package: impactr (via r-universe)

November 1, 2024

Title Mechanical Loading Prediction Through Accelerometer Data

Version 0.4.1

Description Functions to read, process and analyse accelerometer data related to mechanical loading variables. This package is developed and tested for use with raw accelerometer data from triaxial 'ActiGraph' <https://actigraphcorp.com> accelerometers.

License MIT + file LICENSE

URL https://lveras.com/impactr/

BugReports https://github.com/verasls/impactr/issues/

Encoding UTF-8

LazyData true

Roxygen list(markdown = TRUE)

RoxygenNote 7.2.1

Imports glue, lubridate, lvmisc, pillar, pracma, purrr, Rcpp, rlang (>= 0.4.6), signal, stringr, tibble, toOrdinal, utils, vroom

Suggests accdata, covr, knitr, rmarkdown, testthat (>= 3.0.0)

Config/testthat/edition 3

LinkingTo Rcpp

VignetteBuilder knitr

Language en-GB

Depends R (>= 2.10)

Additional_repositories https://lveras.com/drat/

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

RemoteUrl https://github.com/verasls/impactr

RemoteRef HEAD

RemoteSha 30edf5060f2da2703596f528013eeb759d2ae57e

Contents

`define_region` *Define region of interest*

Description

Define the region of interest for data analysis based on the accelerometer data timestamp.

Usage

```
define_region(data, start_time, end_time)
```

Arguments

`data`	An `impactr_data` object, as obtained with read_acc().
`start_time, end_time`	A character string with the start and end times of the region of interest in the "YYYY-MM-DD HH:MM:SS" format.

Value

An object of class `impactr_data`.

Examples

```
data <- read_acc(impactr_example("hip-raw.csv"))
define_region(
  data, start_time = "2021-04-06 15:45:00", end_time = "2021-04-06 15:46:00"
)
```

`filter_acc` *Filter the acceleration signal*

Description

Filter the acceleration signal using a butterworth digital filter.

Usage

```
filter_acc(data, order = 4, cutoff = 20, type = "lowpass")
```

Arguments

`data`	An `impactr_data` object, as obtained with read_acc().
`order`	The order of the filter. Defaults to 4.
`cutoff`	The filter cut-off frequency in Hz. Defaults to 20. For low- and high-pass filters, must be a scalar. For band-pass and band- stop, a vector of length two.
`type`	The type of filter. Defaults to "lowpass". Can be "lowpass", "highpass", "band-pass" or "bandstop".

Details

The default values of the filter parameters are matching the filter used in the paper by Veras et al. that developed the mechanical loading prediction equations (see References).

Value

An object of class `impactr_data`.

References

- Veras L, Diniz-Sousa F, Boppre G, Devezas V, Santos-Sousa H, Preto J, Machado L, Vilas-Boas JP, Oliveira J, Fonseca H. Accelerometer-based prediction of skeletal mechanical loading during walking in normal weight to severely obese subjects. Osteoporosis International. 2020. 31(7):1239- 1250. doi:10.1007/s00198020052952.

Examples

```
data <- read_acc(impactr_example("hip-raw.csv"))
filter_acc(data)
```

`find_peaks` *Find peaks in a signal*

Description

Find peaks in the acceleration signal.

Usage

```
find_peaks(data, vector, min_height = 1.3, min_dist = 0.4)
```

Arguments

`data`	An `impactr_data` object, as obtained with read_acc().
`vector`	A character string indicating in which acceleration vector to find the peaks. Can be "resultant", "vertical" or "all".
`min_height`	The minimum height of the peaks (in *g*).
`min_dist`	The minimum horizontal distance between peaks (in seconds).

Details

The default values of the filter parameters are matching the filter used in the paper by Veras et al. that developed the mechanical loading prediction equations (see References). When the `vector` parameter is set to "all", there may contain `NA` values in the `resultant_peak_acc` and/or `vertical_peak_acc` at the timestamps in which a peak value for that vector could not be identified.

The default values of `min_height` and `min_dist` are matching the criteria used in the paper by Veras et al. that developed the mechanical loading prediction equations (see References)

Value

An object of class `impactr_peaks` with the peaks magnitude stored in the columns.

References

- Veras L, Diniz-Sousa F, Boppre G, Devezas V, Santos-Sousa H, Preto J, Machado L, Vilas-Boas JP, Oliveira J, Fonseca H. Accelerometer-based prediction of skeletal mechanical loading during walking in normal weight to severely obese subjects. Osteoporosis International. 2020. 31(7):1239- 1250. doi:10.1007/s00198020052952.

Examples

```
data <- read_acc(impactr_example("hip-raw.csv"))
data <- use_resultant(data)
find_peaks(data, vector = "resultant")
```

impactr_example *Get path to example data*

Description

impactr comes with some example ActiGraph accelerometer raw data files in its inst/extdata directory. This function make them easy to access.

Usage

```
impactr_example(file = NULL)
```

Arguments

file	A character string with the file name. If NULL, the example files will be listed.

Value

If file = NULL, it returns the file names of the example data files, else it returns the path to the example data.

Examples

```
impactr_example()
impactr_example("hip-raw.csv")
```

import_dataset *Import datasets from accdata package*

Description

A helper function to import datasets from the accdata package.

Usage

```
import_dataset(data)
```

Arguments

data	A character string indicating which data to load. The currently available datasets are "daily_acc_3d" and "daily_acc_7d".

Details

To import these datasets you need to install the accdata package. It can be installed by running install_accdata(). The datasets documentation can be accessed by ?accdata::`dataset_name` (e.g., ?accdata::daily_acc_3d.

Value

An object of class `impactr_data`.

Examples

```
# Ensure that {accdata} package is available before running the example.
# If it is not, run install_accdata() to install the required package.
if (requireNamespace("accdata", quietly = TRUE)) {
  data <- import_dataset("daily_acc_3d")
  data
}
```

`install_accdata` *Install accdata package*

Description

A helper function to install the `accdata` package from a `drat` repository. The `accdata` package contains datasets that can be used to test the functionalities from `impactr`. Note that `accdata` is a large package (approximately 80 MB) and could take a while to download and install.

Usage

```
install_accdata()
```

`is_impactr` *Test if the object is from the impactr package*

Description

Test if the object is from the impactr package

Usage

```
is_impactr_data(x)

is_impactr_peaks(x)
```

Arguments

`x`	An object.

Value

`TRUE` if the object inherits the class being evaluated.

predict_loading *Predict mechanical loading*

Description

Predict either ground reaction force or loading rate, or both, based on accelerometer data.

Usage

```
predict_loading(data, outcome, vector, model)
```

Arguments

data	An impactr_data object, as obtained with read_acc().
outcome	A character string. Can be either "grf" (for ground reaction force), or "lr" (for loading rate) or "all" (for both mechanical loading variables).
vector	A character string indicating in which acceleration vector to find the peaks. Can be "resultant", "vertical" or "all".
model	A character string indicating which model to use to make the predictions. The values currently supported are "walking", "walking/running" and "jumping".

Value

An object of class impactr_peaks with the ground reaction force and/or loading rate peaks magnitude stored in the columns.

Examples

```
data <- read_acc(impactr_example("hip-raw.csv"))
data <- specify_parameters(data, acc_placement = "hip", subj_body_mass = 78)
data <- find_peaks(data, vector = "vertical")
predict_loading(
  data,
  outcome = "grf",
  vector = "vertical",
  model = "walking/running"
)
```

read_acc *Read raw accelerometer data*

Description

Reads raw accelerometer data files into an `impactr_data` object.

Usage

```
read_acc(file)
```

Arguments

`file`	Path to a raw accelerometer data file.

Value

An object of class `impactr_data`.

Examples

```
read_acc(impactr_example("hip-raw.csv"))
```

remove_nonwear *Detect and remove accelerometer non-wear time*

Description

Detects the accelerometer non-wear time based on an algorithm developed by van Hees (see Details) and remove these periods from the raw data. This function can also draw a plot to better visualize the detected non-wear periods and generate a wear time daily summary.

Usage

```
remove_nonwear(
  data,
  window1 = 60,
  window2 = 15,
  threshold = 2,
  min_hour_crit = 0,
  min_day_crit = 0,
  plot = FALSE,
  save_plot = FALSE,
  save_summary = FALSE
)
```

Arguments

`data`	An `impactr_data` object, as obtained with read_acc().
`window1`, `window2`	Windows size, in minutes, for the non-wear detection algorithm. Defaults to 60 and 15 minutes, respectively. Also, `window2` must be smaller than `window1`, and `window1` must be a multiple of `window2`.
`threshold`	Number of axes that need to meet the non-wear criteria. Defaults to 2.
`min_hour_crit`	The minimum number of hours marked as wear time in a day for it to be considered valid (see Data validation). Defaults to 0, meaning that every day is considered valid.
`min_day_crit`	The minimum number of valid days for the data of a given subject to be considered valid (see Data validation). Defaults to 0, meaning that all data is valid.
`plot`	A logical value indicating whether or not to display the plot to visualize the detected non-wear periods. Defaults to `FALSE`. Notice that the plot will only be displayed in your R session if you do not provide a path to save the plot (see the argument `save_plot`).
`save_plot`, `save_summary`	Indicates whether of not to save the plot to visualize the detected non-wear periods to a pdf file and the wear time daily summary to a csv file, respectively. Defaults to `FALSE`. Provide a valid path to a file, ending with the ".pdf" extension for the plot or with the ".csv" extension to the summary, as a character string if you want the outputs to be saved.

Value

An object of class `impactr_data` and a plot if `plot = TRUE` and `save_plot = FALSE`.

The non-wear detection algorithm

The current version of this algorithm is described in a paper by van Hees et al (see References) and also in this vignette from package GGIR. Briefly, in a first stage it identifies non-wear time based on threshold values of standard deviation ($0.013g$) and range ($0.050g$) of raw acceleration from each axis. The classification is done per blocks of `window2` size (default 15 minutes) based on the characteristics of a larger `window1` (default 60 minutes) centred at the `window2`. In the second stage of the algorithm, the plausibility of wear periods in between non-wear periods is tested based on the duration and proportion of the duration relative to the surrounding non-wear periods.

Data validation

After the detection of non-wear periods through the algorithm, a data validation step is applied. For each measurement day to be considered valid, it has to present a minimum number of wear time hours determined by the `min_hour_crit` argument. If the number of wear time hours of a given day falls below the threshold, the whole day is considered invalid and is then removed from the subsequent analyses. The whole measurement is also classified as valid or invalid based on the number of valid days and a threshold given by `min_day_crit`. If the number of valid days is less than the value determined by the `min_day_crit` argument, the whole data is deleted and the `remove_nonwear()` function signals an error, stopping its execution. Nevertheless, this error does

not prevent the plot to be displayed or saved, or the wear time daily summary to be saved, if the arguments are set to do so.

References

- van Hees VT, Gorzelniak L, Dean León EC, Eder M, Pias M, Taherian S, Ekelund U, Renström F, Franks PW, Horsch A, Brage S. Separating movement and gravity components in an acceleration signal and implications for the assessment of human daily physical activity. PLoS One. 2013. Apr 23. doi:10.1371/journal.pone.0061691.

Examples

```
# Ensure that {accdata} package is available before running the example.
# If it is not, run install_accdata() to install the required package.
if (requireNamespace("accdata", quietly = TRUE)) {
  data <- import_dataset("daily_acc_3d")
  remove_nonwear(data)
}
```

`specify_parameters` *Specify prediction model parameters*

Description

Specify the accelerometer placement used and the subject body mass. These data is needed in order to use the mechanical loading prediction models.

Usage

```
specify_parameters(data, acc_placement, subj_body_mass)
```

Arguments

`data`	An `impactr_data` object, as obtained with read_acc().
`acc_placement`	A character string indicating the accelerometer placement. Can be either "ankle", "back", or "hip".
`subj_body_mass`	A double scalar indicating the subject body mass in kilograms.

Value

An object of class `impactr_data` with the specified parameters as attributes.

Examples

```
data <- read_acc(impactr_example("hip-raw.csv"))
specify_parameters(data, acc_placement = "hip", subj_body_mass = 79.2)
```

summarise_loading *Summarise mechanical loading variables*

Description

Creates a summary table of the selected mechanical loading variables including the number of peaks, the minimum, maximum, mean and standard deviation values of these peaks and also the number of peaks inside a given magnitude range. The summaries can be displayed by day or as a daily average.

Usage

```
summarise_loading(
  data,
  variable,
  vector,
  daily_average = TRUE,
  ranges_acc = NULL,
  ranges_grf = NULL,
  ranges_lr = NULL,
  save_summary = FALSE
)
```

Arguments

`data`	An `impactr_peaks` object, as obtained with find_peaks() and/or predict_loading().
`variable`	A character vector indicating the variable to summarise. Can be either "acc" (for the acceleration peaks), "grf" (for the ground reaction force peaks), "lr" (for the loading rate peaks) or "all" (for all variables).
`vector`	A character string indicating which vector to use to create the summaries. Can be "resultant", "vertical" or "all".
`daily_average`	Create a daily average summary? Can be `TRUE` (default) or `FALSE`.
`ranges_acc`, `ranges_grf`, `ranges_lr`	A numeric vector to specify ranges in which to count the peaks. *E.g.*, If `ranges_acc = c(1, 2, 3)`, it will summarise the number of acceleration peaks from 1 to 2g, from 2 to 3g and above 3g. Set to `NULL` (default) if no summary by range will be provided.
`save_summary`	Indicates whether or not to save the summary to a csv file(s). Defaults to `FALSE`. Provide a valid path to a directory as a character string to save all generated summaries.

Value

A tibble (or a list of tibbles) with the requested summaries.

Examples

```
# Ensure that {accdata} package is available before running the example.
# If it is not, run install_accdata() to install the required package.
if (requireNamespace("accdata", quietly = TRUE)) {
  data <- import_dataset("daily_acc_3d")
  data <- remove_nonwear(data)
  data <- filter_acc(data)
  data <- find_peaks(data, vector = "vertical")
  summarise_loading(
    data,
    variable = "acc", vector = "vertical",
    ranges_acc = 1:5
  )
}
```

`use_resultant` *Use resultant vector*

Description

Computes the acceleration resultant vector.

Usage

```
use_resultant(data)
```

Arguments

`data`	An `impactr_data` object, as obtained with

Value

An object of class `impactr_data` with the `acc_R` column containing the acceleration resultant vector.

Examples

```
data <- read_acc(impactr_example("hip-raw.csv"))
use_resultant(data)
```

Index