Package 'wearables'

October 12, 2022

Type Package

Date 2021-12-20

Title Tools to Read and Convert Wearables Data

Version 0.8.1

Imports xts, lubridate, dplyr, RHRV, magrittr, signal, waveslim, futile.logger, kernlab, padr, varian, ggplot2, R.utils

Maintainer Peter de Looff <peterdelooff@gmail.com>

Description Package to read Empatica E4 data, perform several transformations, perform signal processing and analyses, including batch analyses.

License GPL-2

Encoding UTF-8

LazyData true

Depends R (>= 3.6)

RoxygenNote 7.1.2

Suggests testthat

NeedsCompilation no

Author Peter de Looff [aut, cre],
Remko Duursma [aut],
Saskia Koldijk [aut],
Kees de Schepper [aut],
Matthijs Noordzij [ctb],
Natasha Jaques [ctb],
Sara Taylor [ctb]

Repository CRAN

Date/Publication 2021-12-20 15:20:02 UTC

R topics documented:

`add_chunk_group` *Addition of chunk groups*

Description

partition data into chunks of a fixed number of rows in order to calculate aggregated features per chunk

Usage

```
add_chunk_group(data, rows_per_chunk)
```

Arguments

`data`	df to partition into chunks
`rows_per_chunk`	size of a chunk

`aggregate_e4_data` *Aggregate E4 data into 1min timesteps*

Description

Aggregate E4 data into 1min timesteps

Usage

```
aggregate_e4_data(x)
```

Arguments

`x`	An object read by `read_e4`.

as_time *as_time*

Description

Converts Unix time to as.POSIXct

Usage

```
as_time(x, tz = "UTC")
```

Arguments

x	takes a unixtime and converts to as.POSIXct
tz	timezone is set to UTC

as_timeseries *Convert an E4 data stream to a timeseries*

Description

Creates an xts object indexed by time

Usage

```
as_timeseries(data, index = 2, name_col = "V1")
```

Arguments

data	A dataframe, subelements of list as output by read_e4 function
index	Which column (integer) to use as the data in the timeseries. Default: 2.
name_col	Column name to give to the timeseries data.

`batch_analysis` *Batch analysis*

Description

Read and process all ZIP files in a directory

Usage

```
batch_analysis(path_in = NULL, path_out = ".")
```

Arguments

`path_in`	input path
`path_out`	output path

`binary_classifier_config`
Configuration of the SVM algorithm for binary classification

Description

Configuration of the SVM algorithm for binary classification

Usage

```
binary_classifier_config
```

Format

An object of class `list` of length 4.

Author(s)

Sara Taylor <sataylor@mit.edu>

References

https://eda-explorer.media.mit.edu/

`calculate_RMSSD` *RMSSD calculation*

Description

Calculation of RMSSD over 1 minute time periods for plotting

Usage

```
calculate_RMSSD(IBIdata)
```

Arguments

`IBIdata`	Uses the IBI data frame as created by `read_e4`

`char_clock_systime` *Force character datetime variable ("yyyy-mm-dd hh:mm:ss") to system timezone*

Description

Force character datetime variable ("yyyy-mm-dd hh:mm:ss") to system timezone

Usage

```
char_clock_systime(time)
```

Arguments

`time`	Datetime variable ("yyyy-mm-dd hh:mm:ss")

`choose_between_classes` *Choice between two classes*

Description

Make choice between two classes based on kernel values

Usage

```
choose_between_classes(class_a, class_b, kernels)
```

Arguments

class_a	Number by which class a is indicated
class_b	Number by which class b is indicated
kernels	Kernel values from SVM

---

compute_amplitude_features *Amplitude features*

Description

Compute amplitude features.

Usage

```
compute_amplitude_features(data)
```

Arguments

data	vector of amplitude values

---

compute_derivative_features *Derivative features*

Description

Compute derivative features.

Usage

```
compute_derivative_features(derivative, feature_name)
```

Arguments

derivative	vector of derivatives
feature_name	name of feature

`compute_features2` *Features computation*

Description

Compute features for SVM

Usage

```
compute_features2(data)
```

Arguments

`data`	df with eda, filtered eda and timestamp columns

`compute_wavelet_coefficients` *Wavelet coefficients*

Description

Compute wavelet coefficients.

Usage

```
compute_wavelet_coefficients(data)
```

Arguments

`data`	data with an EDA element

`compute_wavelet_decomposition` *Wavelet decomposition*

Description

Compute wavelet decomposition.

Usage

```
compute_wavelet_decomposition(data)
```

Arguments

`data`	vector of values

create_e4_output_folder *Output folder*

Description

Create output folder for E4 analysis results

Usage

```
create_e4_output_folder(obj, out_path = ".")
```

Arguments

obj	e4 analysis object
out_path	output folder

e4_filecut_intervals *Filter datasets for a Datetime start + end*

Description

A function to determine how many intervals should be created. The question is at what time do you want the filecut to start, what should be the period that you want separate files for, and what should the interval be?

Usage

```
e4_filecut_intervals(time_start, time_end, interval)
```

Arguments

time_start	User input start time in the character format "yyyy-mm-dd hh:mm:ss" / e.g., "2019-11-27 08:32:00". Where do you want the file cut to start?
time_end	User input end time (same format as time_start)
interval	# Interval: User input interval (in minutes/ e.g., 5) What is the duration of the interval you want to divide the period into? For example, the paper by de Looff et al. (2019) uses 5 minute intervals over a 30 minute period preceding aggressive behavior. The 5 minute interval is chosen as for the calculation of some of the heart rate variability parameters one needs at least 5 minutes of data, but shorter intervals are possible as well, see for instance: Shaffer, Fred, en J. P. Ginsberg. 'An Overview of Heart Rate Variability Metrics and Norms'. Frontiers in Public Health 5 (28 september 2017). https://doi.org/10.3389/fpubh.2017.00258.

`filter_createdir_zip` *Function to filter the data object based on the time period and intervals that are needed for the files to be cut. The function also creates identical Empatica E4 zipfiles in the same directory as where the original zipfile is located.*

Description

Function to filter the data object based on the time period and intervals that are needed for the files to be cut. The function also creates identical Empatica E4 zipfiles in the same directory as where the original zipfile is located.

Usage

```
filter_createdir_zip(
  data,
  time_start,
  time_end,
  interval,
  out_path = NULL,
  fn_name = NULL
)
```

Arguments

`data`	Object read with `read_e4`
`time_start`	User input start time in the character format "yyyy-mm-dd hh:mm:ss" / e.g., "2019-11-27 08:32:00". Where do you want the file cut to start?
`time_end`	User input end time (same format as time_start)
`interval`	# Interval: User input interval (in minutes/ e.g., 5) What is the duration of the interval you want to divide the period into? For example, the paper by de Looff et al. (2019) uses 5 minute intervals over a 30 minute period preceding aggressive behavior. The 5 minute interval is chosen as for the calculation of some of the heart rate variability parameters one needs at least 5 minutes of data.
`out_path`	The directory where to write the cut files; defaults to the input folder.
`fn_name`	The directory where to write the cut files without the extension.

Value

out_path fn_name

filter_e4data_datetime

Filter all four datasets for a Datetime start + end

Description

Filter all four datasets for a Datetime start + end

Usage

```
filter_e4data_datetime(data, start, end)
```

Arguments

data	Object read with read_e4
start	Start Datetime (posixct)
end	End Datetime (posixct)

find_peaks

Function to find peaks of an EDA datafile

Description

This function finds the peaks of an EDA signal and adds basic properties to the datafile.

Usage

```
find_peaks(
  data,
  offset = 1,
  start_WT = 4,
  end_WT = 4,
  thres = 0.005,
  sample_rate = getOption("SAMPLE_RATE", 8)
)
```

Arguments

data	DataFrame with EDA as one of the columns and indexed by a datetimeIndex
offset	the number of rising seconds and falling seconds after a peak needed to be counted as a peak
start_WT	maximum number of seconds before the apex of a peak that is the "start" of the peak

end_WT	maximum number of seconds after the apex of a peak that is the "end" of the peak 50 percent of amp
thres	the minimum microsecond change required to register as a peak, defaults as .005
sample_rate	number of samples per second, default=8

Details

Also, peak_end is assumed to be no later than the start of the next peak. Is that OK?

Value

data frame with several columns peaks 1 if apex peak_start 1 if start of peak peak_end 1 if end of preak peak_start_times if apex then corresponding start timestamp peak_end_times if apex then corresponding end timestamp half_rise if sharp decaying apex then time to halfway point in rise amp if apex then value of EDA at apex - value of EDA at start max_deriv if apex then max derivative within 1 second of apex rise_time if apex then time from start to apex decay_time if sharp decaying apex then time from apex to end SCR_width if sharp decaying apex then time from half rise to end

---

get_amp *Peak amplitude*

Description

Get the amplitude of the peaks

Usage

```
get_amp(data)
```

Arguments

data	df with peak info

---

get_apex *Get the eda apex of the signal*

Description

finds the apex of electrodermal activity eda signal within an optional time window

Usage

```
get_apex(eda_deriv, offset = 1)
```

Arguments

eda_deriv	uses the eda derivative to find the apex
offset	minimum number of downward measurements after the apex, in order to be considered a peak (default 1 means no restrictions)

---

get_decay_time *Decay time*

Description

Get the time (in seconds) it takes to decay for each peak

Usage

```
get_decay_time(data, i_apex_with_decay)
```

Arguments

data	df with peak info
i_apex_with_decay	indexes of relevant peaks

---

get_derivative *First derivative*

Description

Get the first derivative.

Usage

```
get_derivative(values)
```

Arguments

values	vector of numbers

get_eda_deriv *Electrodermal activity signal derivative*

Description

Finds the first derivatives of the eda signal

Usage

```
get_eda_deriv(eda)
```

Arguments

eda	eda vector

get_half_amp *Half peak amp*

Description

Get the amplitude value halfway between peak start and apex

Usage

```
get_half_amp(data, i)
```

Arguments

data	df with peak info
i	apex index

get_half_rise *Half rise time*

Description

Get the time (in seconds) it takes to get to halfway the rise in a peak

Usage

```
get_half_rise(data, i_apex_with_decay)
```

Arguments

data	df with peak info
i_apex_with_decay	relevant apices

get_i_apex_with_decay *Decaying peaks*

Description

Identify peaks with a decent decay (at least half the amplitude of rise)

Usage

```
get_i_apex_with_decay(data)
```

Arguments

data	df with peak info

get_kernel *SVM kernel*

Description

Generate kernel needed for SVM

Usage

```
get_kernel(kernel_transformation, sigma, columns)
```

Arguments

kernel_transformation	Data matrix used to transform EDA features into kernel values
sigma	The inverse kernel width used by the kernel
columns	Features computed from EDA signal

get_max_deriv *Maximum derivative*

Description

Get the largest slope before apex, interpolated to seconds

Usage

```
get_max_deriv(data, eda_deriv, sample_rate)
```

Arguments

data	df with info on the peaks
eda_deriv	derivative of the signal
sample_rate	sample rate of the signal

get_peak_end *Peak end*

Description

Find the end of the peaks, with some restrictions on the search

Usage

```
get_peak_end(data, max_lookahead)
```

Arguments

data	df with peak info
max_lookahead	max distance from apex to search for end

get_peak_end_times *Peak end times*

Description

Get the end timstamp of the peaks

Usage

```
get_peak_end_times(data)
```

Arguments

data	df with peak info

get_peak_start *Start of peaks*

Description

Provide info for each measurement whether it is the start of a peak (0 or 1)

Usage

```
get_peak_start(data, sample_rate)
```

Arguments

data	df with peak info
sample_rate	sample rate of the signal

get_peak_start_times *Peak start times*

Description

Get the start times of the peaks

Usage

```
get_peak_start_times(data)
```

Arguments

data	df with peak info

get_rise_time *Rise time of peaks*

Description

Calculates the rise time of all peaks

Usage

```
get_rise_time(eda_deriv, apices, sample_rate, start_WT)
```

Arguments

eda_deriv	first derivative of signal
apices	apex status per measurement (0 or 1)
sample_rate	sample rate of the signal
start_WT	window within which to look for rise time (in seconds)

---

get_SCR_width *Peak width*

Description

Get the width of the peak (in seconds, from halfway the rise until the end)

Usage

```
get_SCR_width(data, i_apex_with_decay)
```

Arguments

data	df with peak info
i_apex_with_decay	relevant apices

---

get_second_derivative *Second derivative*

Description

Get the second derivative.

Usage

```
get_second_derivative(values)
```

Arguments

values	vector of numbers

ibi_analysis *IBI analysis*

Description

Analysis of interbeat interval (IBI)

Usage

```
ibi_analysis(IBI)
```

Arguments

IBI	IBI data, component of object (the number of seconds since the start of the recording) read with read_e4

max_per_n *Max value per segment of length n*

Description

Give the maximum value of a vector of values per segment of length n.

Usage

```
max_per_n(values, n, output_length)
```

Arguments

values	array of numbers
n	length of each segment
output_length	argument to adjust for final segment not being full

multiclass_classifier_config *Configuration of the SVM algorithm for ternary classification*

Description

Configuration of the SVM algorithm for ternary classification

Usage

```
multiclass_classifier_config
```

Format

An object of class `list` of length 4.

Author(s)

Sara Taylor <sataylor@mit.edu>

References

https://eda-explorer.media.mit.edu/

pad_e4 *pad_e4*

Description

function to combine several e4 files, and sets the length of the x-axis

Usage

```
pad_e4(x)
```

Arguments

x	index of dataframe

`plot_artifacts` *Artifact plots*

Description

Plot artifacts after eda_data is classified

Usage

```
plot_artifacts(labels, eda_data)
```

Arguments

`labels`	labels with artifact classification
`eda_data`	data upon which the labels are plotted

`predict_binary_classifier` *Binary classifiers*

Description

Generate classifiers (artifact, no artifact)

Usage

```
predict_binary_classifier(data)
```

Arguments

`data`	features from EDA signal

`predict_multiclass_classifier` *Ternary classifiers*

Description

Generate classifiers (artifact, unclear, no artifact)

Usage

```
predict_multiclass_classifier(data)
```

Arguments

`data`	features from EDA signal

prepend_time_column *prepend_time_column*

Description

Column binds a time_column to the dataframe

Usage

```
prepend_time_column(data, timestart, hertz, tz = Sys.timezone())
```

Arguments

data	dataframe
timestart	the start of the recording
hertz	hertz in which the E4 data was recorded
tz	The timezone, defaults to user timezone

print.e4data *Show class of object*

Description

Returns 'object of class'

Usage

```
## S3 method for class 'e4data'
print(x, ...)
```

Arguments

x	An e4 data list
...	Further arguments currently ignored.

process_eda *Process EDA data*

Description

Process EDA data

Usage

```
process_eda(eda_data)
```

Arguments

eda_data	Data read with read_e4

rbind_e4 *Row-bind E4 datasets*

Description

Row-bind E4 datasets

Usage

```
rbind_e4(data)
```

Arguments

data	An object read in by read_e4

read_and_process_e4 *Read, process and feature extraction of E4 data*

Description

Reads the raw ZIP file using ‘read_e4‘, performs analyses with ‘ibi_analysis‘ and ‘eda_analysis‘.

Usage

```
read_and_process_e4(zipfile, tz = Sys.timezone())

process_e4(data)
```

Arguments

zipfile	zip file with e4 data to be read
tz	timezone where data were recorded (default system timezone)
data	object from read_e4 function

Value

An object with processed data and analyses, object of class 'e4_analysis'.

---

read_e4 *Read E4 data*

Description

Reads in E4 data as a list (with EDA, HR, Temp, ACC, BVP, IBI as dataframes), and prepends timecolumns

Usage

```
read_e4(zipfile = NULL, tz = Sys.timezone())
```

Arguments

zipfile	A zip file as exported by the instrument
tz	The timezone used by the instrument (defaults to user timezone).

Details

This function reads in a zipfile as exported by Empatica Connect. Then it extracts the zipfiles in a temporary folder and unzips the csv files in the temporary folder.

The EDA, HR, BVP, and TEMP csv files have a similar structure in which the starting time of the recording is read from the first row of the file (in unix time). The frequency of the measurements is read from the second row of the recording (in Hz). Subsequently, the raw data is read from row three onward.

The ACC csv file contain the acceleration of the Empatica E4 on the three axes x,y and z. The first row contains the starting time of the recording in unix time. The second row contains the frequency of the measurements in Hz. Subsequently, the raw x, y, and z data is read from row three onward.

The IBI file has a different structure, the starting time in unix is in the first row, first column. The firs column contins the number of seconds past since the start of the recording. The number of seconds past since the start of the recording represent a heartbeat as derived from the algorithms from the photo plethysmogrophy sensor. The second column contains the duration of the interval from one heartbeat to the next heartbeat.

ACC.csv = 32 Hz BVP.csv = 64 Hz EDA.csv = 4 HZ HR.csv = 1 HZ TEMP.csv = 4 Hz

Please also see the info.txt file provided in the zip file for additional information.

The function returns an object of class "e4_data" with a prepended datetime columns that defaults to user timezone. The object contains a list with dataframes from the physiological signals.

Examples

```
library(wearables)
#read_e4()
```

`remove_small_peaks` *Small peaks removal*

Description

Remove peaks with a small rise from start to apex are removed

Usage

```
remove_small_peaks(data, thres = 0)
```

Arguments

`data`	df with info on peaks
`thres`	threshold of amplitude difference in order to be removed (default 0 means no removals)

`upsample_data_to_8Hz` *Upsample EDA data to 8 Hz*

Description

Upsample EDA data to 8 Hz

Usage

```
upsample_data_to_8Hz(eda_data)
```

Arguments

`eda_data`	Data read with `read_e4`

`write_processed_e4` *Write CSV files of the output*

Description

Slow!

Usage

```
write_processed_e4(obj, out_path = ".")
```

Arguments

`obj`	e4 analysis object
`out_path`	output folder

Index