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 .................................................. 3
  aggregate_e4_data ............................................... 3
## R topics documented:

<table>
<thead>
<tr>
<th>Function</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>as_time</td>
<td>4</td>
</tr>
<tr>
<td>as_timeseries</td>
<td>4</td>
</tr>
<tr>
<td>batch_analysis</td>
<td>5</td>
</tr>
<tr>
<td>binary_classifier_config</td>
<td>5</td>
</tr>
<tr>
<td>calculate_RMSSD</td>
<td>6</td>
</tr>
<tr>
<td>char_clock_systime</td>
<td>6</td>
</tr>
<tr>
<td>choose_between_classes</td>
<td>6</td>
</tr>
<tr>
<td>compute_amplitude_features</td>
<td>7</td>
</tr>
<tr>
<td>compute_derivative_features</td>
<td>7</td>
</tr>
<tr>
<td>compute_features2</td>
<td>8</td>
</tr>
<tr>
<td>compute_wavelet_coefficients</td>
<td>8</td>
</tr>
<tr>
<td>compute_wavelet_decomposition</td>
<td>8</td>
</tr>
<tr>
<td>create_e4_output_folder</td>
<td>9</td>
</tr>
<tr>
<td>e4_filecut_intervals</td>
<td>9</td>
</tr>
<tr>
<td>filter_createdir_zip</td>
<td>10</td>
</tr>
<tr>
<td>filter_e4data_datetime</td>
<td>11</td>
</tr>
<tr>
<td>find_peaks</td>
<td>11</td>
</tr>
<tr>
<td>get_amp</td>
<td>12</td>
</tr>
<tr>
<td>get_apex</td>
<td>12</td>
</tr>
<tr>
<td>get_decay_time</td>
<td>13</td>
</tr>
<tr>
<td>get_derivative</td>
<td>13</td>
</tr>
<tr>
<td>get_eda_deriv</td>
<td>14</td>
</tr>
<tr>
<td>get_half_amp</td>
<td>14</td>
</tr>
<tr>
<td>get_half_rise</td>
<td>14</td>
</tr>
<tr>
<td>get_i_apex_with_decay</td>
<td>15</td>
</tr>
<tr>
<td>get_kernel</td>
<td>15</td>
</tr>
<tr>
<td>get_max_deriv</td>
<td>16</td>
</tr>
<tr>
<td>get_peak_end</td>
<td>16</td>
</tr>
<tr>
<td>get_peak_end_times</td>
<td>16</td>
</tr>
<tr>
<td>get_peak_start</td>
<td>17</td>
</tr>
<tr>
<td>get_peak_start_times</td>
<td>17</td>
</tr>
<tr>
<td>get_rise_time</td>
<td>17</td>
</tr>
<tr>
<td>get_SCR_width</td>
<td>18</td>
</tr>
<tr>
<td>get_second_derivative</td>
<td>18</td>
</tr>
<tr>
<td>ibi_analysis</td>
<td>19</td>
</tr>
<tr>
<td>max_per_n</td>
<td>19</td>
</tr>
<tr>
<td>multiclass_classifier_config</td>
<td>20</td>
</tr>
<tr>
<td>pad_e4</td>
<td>20</td>
</tr>
<tr>
<td>plot_artifacts</td>
<td>21</td>
</tr>
<tr>
<td>predict_binary_classifier</td>
<td>21</td>
</tr>
<tr>
<td>predict_multiclass_classifier</td>
<td>21</td>
</tr>
<tr>
<td>prepend_time_column</td>
<td>22</td>
</tr>
<tr>
<td>print.e4data</td>
<td>22</td>
</tr>
<tr>
<td>process_eda</td>
<td>23</td>
</tr>
<tr>
<td>rbind_e4</td>
<td>23</td>
</tr>
<tr>
<td>read_and_process_e4</td>
<td>23</td>
</tr>
<tr>
<td>read_e4</td>
<td>24</td>
</tr>
<tr>
<td>remove_small_peaks</td>
<td>25</td>
</tr>
</tbody>
</table>
add_chunk_group

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

Description
Aggregate E4 data into 1min timesteps

Usage
aggregate_e4_data(x)

Arguments
- x: An object read by read_e4.
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**

<table>
<thead>
<tr>
<th>Description</th>
<th><strong>Batch analysis</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Read and process all ZIP files in a directory</td>
<td></td>
</tr>
</tbody>
</table>

**Usage**

```r
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*

<table>
<thead>
<tr>
<th>Description</th>
<th>Configuration of the SVM algorithm for binary classification</th>
</tr>
</thead>
</table>

**Usage**

```r
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/](https://eda-explorer.media.mit.edu/)
choose_between_classes


calculate_RMSSD

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

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

Description
Make choice between two classes based on kernel values

Usage
choose_between_classes(class_a, class_b, kernels)
**compute_amplitude_features**

**Arguments**
- `class_a`: Number by which class a is indicated
- `class_b`: Number by which class b is indicated
- `kernels`: Kernel values from SVM

**Description**
Compute amplitude features.

**Usage**
`compute_amplitude_features(data)`

**Arguments**
- `data`: vector of amplitude values

---

**compute_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

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>obj</td>
<td>e4 analysis object</td>
</tr>
<tr>
<td>out_path</td>
<td>output folder</td>
</tr>
</tbody>
</table>

create_e4_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

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>time_start</td>
<td>User input start time in the character format &quot;yyyymm-dd hh:mm:ss&quot; / e.g., &quot;2019-11-27 08:32:00&quot;. Where do you want the file cut to start?</td>
</tr>
<tr>
<td>time_end</td>
<td>User input end time (same format as time_start)</td>
</tr>
<tr>
<td>interval</td>
<td># 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). <a href="https://doi.org/10.3389/fpubh.2017.00258">https://doi.org/10.3389/fpubh.2017.00258</a>.</td>
</tr>
</tbody>
</table>


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

```r
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 "yyyymm-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 peak 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)
get_decay_time

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_edu_deriv | Electrodermal activity signal derivative

Description
Finds the first derivatives of the eda signal

Usage
get_edu_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**
```python
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**
```python
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 timestamp of the peaks

**Usage**
```python
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)
```
**get_second_derivative**

**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
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/](https://eda-explorer.media.mit.edu/)

---

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

### 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

### Description

Returns 'object of class'

### Usage

```r
## 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 first column contains the number of seconds past since the start of the recording. 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.
remove_small_peaks

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

data_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
<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>obj</td>
<td>e4 analysis object</td>
</tr>
<tr>
<td>out_path</td>
<td>output folder</td>
</tr>
</tbody>
</table>
Index

* datasets
  - binary_classifier_config, 5
  - multiclass_classifier_config, 20
add_chunk_group, 3
aggregate_e4_data, 3
as_time, 4
as_timeseries, 4
batch_analysis, 5
binary_classifier_config, 5
calculate_RMSSD, 6
char_clock_systime, 6
choose_between_classes, 6
compute_amplitude_features, 7
compute_derivative_features, 7
compute_features2, 8
compute_wavelet_coefficients, 8
compute_wavelet_decomposition, 8
create_e4_output_folder, 9
e4_filecut_intervals, 9
filter_createdir_zip, 10
filter_e4data_datetime, 11
find_peaks, 11
get_amp, 12
get_apex, 12
get_decay_time, 13
get_derivative, 13
get_eda_deriv, 14
get_half_amp, 14
get_half_rise, 14
get_i_apex_with_decay, 15
get_kernel, 15
get_max_deriv, 16
get_peak_end, 16
get_peak_end_times, 16
get_peak_start, 17
get_peak_start_times, 17
get_rise_time, 17
get_SCR_width, 18
get_second_derivative, 18
ibi_analysis, 19
max_per_n, 19
multiclass_classifier_config, 20
pad_e4, 20
plot_artifacts, 21
predict_binary_classifier, 21
predict_multiclass_classifier, 21
prepend_time_column, 22
print.e4data, 22
process_e4(read_and_process_e4), 23
process_eda, 23
rbind_e4, 23
read_and_process_e4, 23
read_e4, 3, 6, 10, 11, 19, 23, 24, 25
remove_small_peaks, 25
upsample_data_to_8Hz, 25
write_processed_e4, 26