

Package ‘sleepr’

November 25, 2017

Title Analyse Activity and Sleep Behaviour

Date 2017-08-17

Version 0.0.0.9000

Description Use behavioural variables to score activity and infer sleep from bouts of immobility.

Depends R (>= 3.00),
behavr

Imports data.table

Suggests testthat,
covr,
knitr

License GPL-3

Encoding UTF-8

LazyData true

URL <https://github.com/rethomics/sleepr>

BugReports <https://github.com/rethomics/sleepr/issues>

RoxygenNote 6.0.1

Roxygen list(markdown = TRUE)

R topics documented:

bout_analysis	2
motion_detectors	3
sleep_annotation	4
Index	6

bout_analysis	<i>Find "bouts" in categorical time series</i>
---------------	--

Description

This function is used to find contiguous regions of unique value in a – potentially irregular/heterogeneous – univariate categorical time series.

Usage

```
bout_analysis(var, data)
```

Arguments

var	name of the variable to use from data
data	data.table containing behavioural variable from or one multiple animals. When it has a key, unique values, are assumed to represent unique individuals (e.g. in a behavr table). Otherwise, it analysis the data as coming from a single animal. data must have a column t representing time.

Value

an object of the same type as data (i.e. [data.table::data.table](#) or [behavr::behavr](#)). Each row is a specific bout characterised by three columns.

- t – its *onset*
- duration – its length
- <var> – a column with the same name as var. The value of var for this bout.

See Also

[todo](#)

Examples

```
#TODO
```

motion_detectors	<i>Motion detector for Ethoscope data</i>
------------------	---

Description

Defines whether a *single animal* is moving according to:

Usage

```
max_velocity_detector(data, time_window_length,
    velocity_correction_coef = 0.003, masking_duration = 6)

max_velocity_detector_legacy(data, velocity_threshold = 0.006)

virtual_beam_cross_detector(data, time_window_length)
```

Arguments

data	data.table::data.table containing behavioural variables of <i>a single animal</i> (no id). It must have the columns xy_dist_log10x1000(for computing subpixel velocity), x(beam cross), t and has_interacted (whether a stimulus was delivered).
time_window_length	number of seconds to be used by the motion classifier. This corresponds to the sampling period of the output data.
velocity_correction_coef	an empirical coefficient to correct velocity with respect to variable framerate.
masking_duration	number of second during which any movement is ignored (velocity is set to 0) after a stimulus is delivered (aka interaction).
velocity_threshold	uncorrected velocity above which an animal is classified as ‘moving’ (for the legacy version).

Details

- Validated and corrected subpixel velocity ([max_velocity_detector](#)), the most rigorous
- Uncorrected subpixel velocity ([max_velocity_detector_legacy](#))
- Crossing a virtual beam in the middle of the region of interest ([virtual_beam_cross_detector](#))

[max_velocity_detector](#) is the default movement classification for real-time ethoscope experiments. It is benchmarked against human-generated ground truth.

These functions are *rarely called directly*, but typically used is in the context of [sleep_annotation](#).

Value

an object of the same type as data (i.e. `data.table::data.table` or `behavr::behavr`) with additional columns:

- `moving` Logical, TRUE iff. motion was detected.
- `beam_crosses` The number of beam crosses (when the animal crosses $x = 0.5$ – that is the midpoint of the region of interest) within the time window
- `max_velocity` The maximal velocity within the time window. The resulting data is sampled at a period equals to `time_window_length`.

See Also

TODO

- [sleep_annotation](#) – which requires a motion detector

sleep_annotation	<i>Score sleep behaviour from immobility</i>
------------------	--

Description

This function first uses a motion classifier to decide whether an animal is moving during a given time window. Then, it defines sleep as contiguous immobility for a minimal duration.

Usage

```
sleep_annotation(data, time_window_length = 10, min_time_immobile = 300,
  motion_detector_FUN = max_velocity_detector, ...)
```

```
sleep_dam_annotation(data, time_window_length = 60, min_time_immobile = 300)
```

Arguments

<code>data</code>	<code>data.table</code> containing behavioural variable from or one multiple animals. When it has a key, unique values, are assumed to represent unique individuals (e.g. in a <code>behavr</code> table). Otherwise, it analysis the data as coming from a single animal. data must have a column <code>t</code> representing time.
<code>time_window_length</code>	number of seconds to be used by the motion classifier. This corresponds to the sampling period of the output data.
<code>min_time_immobile</code>	Minimal duration (in s) of a sleep bout. Immobility bouts longer or equal to this value are considered as sleep.
<code>motion_detector_FUN</code>	function used to classify movement
<code>...</code>	extra arguments to be passed to <code>motion_classifier_FUN</code> .

Details

The default `time_window_length` is 300 seconds also known as the "5 minute rule". `sleep_annotation` is typically used for ethoscope data, whilst `sleep_dam_annotation` only works on DAM2 data. These functions are *rarely used directly*, but rather passed as an argument to a data loading function, so that analysis can be performed on the go.

Value

a [behavr](#) table similar to data with additional variables/annotations (i.e. moving and asleep). The resulting data will only have one data point every `time_window_length` seconds.

See Also

- [motion_detectors](#) – options for the `motion_detector_FUN` argument
- [bout_analysis](#) – to further analyse sleep bouts in terms of onset and length

Examples

```
#todo
```

Index

behavr, [2](#), [4](#), [5](#)
behavr::behavr, [2](#), [4](#)
bout_analysis, [2](#), [5](#)

data.table, [2](#), [4](#)
data.table::data.table, [2–4](#)

max_velocity_detector, [3](#)
max_velocity_detector
 (motion_detectors), [3](#)
max_velocity_detector_legacy, [3](#)
max_velocity_detector_legacy
 (motion_detectors), [3](#)
motion_detectors, [3](#), [5](#)

sleep_annotation, [3](#), [4](#), [4](#)
sleep_dam_annotation
 (sleep_annotation), [4](#)

virtual_beam_cross_detector, [3](#)
virtual_beam_cross_detector
 (motion_detectors), [3](#)