Package 'sleepr'

July 4, 2018

outj 1, = 010
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
Remotes rethomics/behavr
License GPL-3
Encoding UTF-8
LazyData true
<pre>URL https://github.com/rethomics/sleepr</pre>
BugReports https://github.com/rethomics/sleepr/issues
RoxygenNote 6.0.1
Roxygen list(markdown = TRUE)
R topics documented:
bout_analysis
Index

bout_analysis

bout_analysis

Find "bouts" in categorical time series

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 inviduals (e.g. in a behavr table). Otherwise, it analysis the data as comming 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.

References

• The relevant rethomic tutorial section – on sleep analysis

See Also

- sleep_annotation to generate a binary sleep variable
- rle run length encoding function on which this analysis is based

Examples

```
# Bout analysis on binary variable:
dt <- toy_dam_data()
dt[, moving := activity > 0]
bdt <- bout_analysis(moving,dt)
print(bdt)
# With multiple states
dt <- toy_ethoscope_data()
# we discretise x position in three states: left, middle and right (1/3 each)</pre>
```

curate_dead_animals 3

curate_dead_animals

Remove - irrelevant - data after individual have died

Description

This function detects when individuals have died base on their first (very) long bout of immobility. Following data (which may include spurious artefact of movement) are removed.

Usage

```
curate_dead_animals(data, moving_var = moving, time_window = hours(24),
    prop_immobile = 0.01, resolution = 24)
```

Arguments

data	data.table containing behavioural variable from or one multiple animals. When it has a key, unique values, are assumed to represent unique inviduals (e.g. in a behavr table). Otherwise, it analysis the data as comming from a single animal. data must have a column t representing time.
moving_var	logical variable in data used to define the moving (alive) state (default is moving)
time_window	window during which to define death (default is one day)
<pre>prop_immobile</pre>	proportion of immobility that counts as "dead" during time_window (see details)
resolution	how much scanning windows overlap. Expressed as a factor (see details).

Details

This function scans the time series looking for "death" in the right (future) data, within time_window. Death is defined as mean(moving_var) < prop_immobile within a time window. Moving window start every time_window/resolution. resolution = 1 is fast but means no overlap. The default would score an animal as dead it does not move more than *one percent of the time* for at least *one day*. All data following "death" are removed.

Value

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

References

• The relevant rethomic tutorial section – on high-resolution survival analysis

4 motion_detectors

See Also

• sleep_annotation – to score movement and sleep

Examples

```
dt1 <- toy_activity_data()</pre>
#all movement after day 3 is set at 0
dt1[t > days(3), moving := FALSE]
# one artefact of movement is detected at day 3.5
dt1[t == days(3.5), moving := TRUE]
dt2 <- curate_dead_animals(dt1)</pre>
dt3 <- curate_dead_animals(dt1,prop_immobile = 0)</pre>
## Not run:
library(ggplot2)
ggplot(data=dt1[,test:=1],aes(t, as.numeric(moving))) +
  geom_line(data=dt1[,test:=1]) +
  geom_line(data=dt2[, test:=2])+
  geom_line(data=dt3[, test:=3])+
  facet_grid(test ~ .)+
  scale_x_time()
## End(Not run)
```

motion_detectors

Motion detector for Ethocope data

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 *a single animal* (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.

sleep_annotation 5

```
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

• sleep_annotation – which requieres a motion detector

sleep_annotation

Score sleep behaviour from immobility

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, min_time_immobile = 300)
```

6 sleep_annotation

Arguments

data

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

time_window_length

number of seconds to be used by the motion classifier. This corresponds to the sampling period of the output data.

min_time_immobile

Minimal duration (in s) of a sleep bout. Immobility bouts longer or equal to this value are considered as sleep.

motion_detector_FUN

function used to classify movement

... extra arguments to be passed to motion_classifier_FUN.

Details

The default time_window_length is 300 seconds – it is 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.

References

• The relevant rethomic tutorial section – on sleep analysis

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

sleep_annotation 7

```
###### Ethoscope, virutal beam cross classification ########
sleep_dt2 <- sleep_annotation(dt_one_animal,</pre>
                             motion_detector_FUN=virtual_beam_cross_detector)
## Not run:
library(ggplot2)
ggplot(sleep_dt2, aes(t,y="Animal 1",fill=asleep)) +
                                    geom_tile() + scale_x_time()
## End(Not run)
###### DAM data, de facto beam cross classification ######
dt_one_animal <- toy_dam_data(seed=7)</pre>
sleep_dt <- sleep_dam_annotation(dt_one_animal)</pre>
## Not run:
library(ggplot2)
ggplot(sleep_dt, aes(t,y="Animal 1",fill=asleep)) +
                                    geom_tile() + scale_x_time()
## End(Not run)
```

Index

```
behavr, 2, 3, 6
behavr::behavr, 2, 3, 5
bout_analysis, 2, 6
curate\_dead\_animals, 3
data.table, 2, 3, 6
data.table::data.table, 2-5
max_velocity_detector, 5
max_velocity_detector
        (motion_detectors), 4
max_velocity_detector_legacy, 5
{\tt max\_velocity\_detector\_legacy}
        (motion_detectors), 4
motion_detectors, 4, 6
rle, 2
sleep_annotation, 2, 4, 5, 5
sleep\_dam\_annotation
        (sleep_annotation), 5
virtual_beam_cross_detector, 5
virtual_beam_cross_detector
        (motion_detectors), 4
```