Package ‘triact’

January 11, 2023

Type Package

Title Analyzing the Lying Behavior of Cows from Accelerometer Data

Version 0.2.0

Description Assists in analyzing the lying behavior of cows from raw data recorded with a triaxial accelerometer attached to the hind leg of a cow. Allows the determination of common measures for lying behavior including total lying duration, the number of lying bouts, and the mean duration of lying bouts. Further capabilities are the description of lying laterality and the calculation of proxies for the level of physical activity of the cow. Reference: Simmler M., Brouwers S. (2023) <https://gitlab.com/AgroSimi/triact_manuscript>.

License GPL (>= 3)

Encoding UTF-8

LazyData true

Depends R (>= 4.1)

Imports R6 (>= 2.5.0), data.table (>= 1.14.0), parallel (>= 4.1.2), checkmate (>= 2.0.0), lubridate (>= 1.7.10), methods, stats

Suggests signal (>= 0.7-7), tibble, rmarkdown, knitr

VignetteBuilder knitr

NeedsCompilation no

Author Michael Simmler [aut, cre] (<https://orcid.org/0000-0002-4095-4111>), Stijn Brouwers [ctb] (<https://orcid.org/0000-0002-7028-7823>)

Maintainer Michael Simmler <michael.simmler@agroscope.admin.ch>

Repository CRAN

Date/Publication 2023-01-11 06:40:02 UTC

R topics documented:

`bout_summary` *Output of Triact$summarize_bouts()*

Description

Output of Triact$summarize_bouts(). The information contained in the output table depends on the analyses you added to the 'Triact' object using the $add_activity(), $add_lying(), and $add_side() methods.

The asterisk (*) in the column names below stands for one of 'L1', 'L2', 'AdjL1', and 'AdjL2'. DBA is the abbreviation for the dynamic body acceleration.

id Cow id

bout_nr Sequential numbering of the bouts per cow id

startTime Start time of the bout

endTime End time of the bout

duration Duration of the bout (units as specified via the `duration_units` argument)

lying `TRUE` for lying bouts, `FALSE` for standing bouts

side `"L"` for left lying side, `"R"` for right lying side (NA for standing bouts)

mean*DBA Mean of the DBA-based proxy for physical activity

mean*Jerk Mean of the Jerk-based proxy for physical activity

`cows_5hz` *cow acceleration data*

Description

Data collected with a triaxial accelerometers (MSR145, MSR Electronics, Switzerland) attached to the left hind leg of two dairy cows (cow01, cow02). The accelerometer sampling frequency was 5 Hz.

colname	*type*	*description*
id	Factor	unique id for the cow
time	POSIXct	timestamp
acc_fwd	numeric	acceleration from *forward* axis (units: *g*)
acc_up	numeric	acceleration from *up* axis (units: *g*)
acc_right	numeric	acceleration from *right* axis (units: *g*)

Usage

```
cows_5hz
```

Note

From the raw data files distributed with the triact package, cows_5hz can be reproduced as follows:

```
# create a Triact object
my_triact <- Triact$new()

dir <- system.file("extdata", package = "triact")

my_triact$load_files(input = dir,
                     id_substring = c(1, 5),
                     timeFwdUpRight_cols = c(1, 2 ,3, 4),
                     skip = "*DATA")

cows_5hz_recreated <- my_triact$data

# test whether they are identical
identical(cows_5hz_recreated, cows_5hz)
```

Source

Agroscope, 8356 Ettenhausen, Switzerland

---

interval_summary   *Output of Triact$summarize_intervals()*

Description

Output of Triact$summarize_intervals(). The information contained in the output table depends on the arguments bouts and side and on the analyses you added to the 'Triact' object using the $add_activity(), $add_lying(), and $add_side() methods.

The asterisk (*) in the column names below stands for one of 'L1', 'L2', 'AdjL1', and 'AdjL2'. DBA is the abbreviation for the dynamic body acceleration.

startTime: Start time of the interval

centerTime: Center time of the interval (convenient for plotting)

endTime: End time of the interval

duration: Duration of data recordings in the interval. Helpful for identifying incompletely observed intervals at start and end of the recording. Units: As specified via the duration_units argument.

durationStanding: Duration in upright posture. Units: As specified via the duration_units argument.

durationLying: Duration in lying posture. Units: As specified via the duration_units argument.

durationLyingLeft: Duration in lying posture with lying side left. Units: As specified via the duration_units argument.

durationLyingRight: Duration in lying posture with lying side right. Units: As specified via the `duration_units` argument.

mean*DBA: Mean of the DBA-based proxy for physical activity. Units: g

mean*Jerk:Mean of the Jerk-based proxy for physical activity. Units: gs^{-1}

mean*DBAStanding: Mean of the DBA-based proxy for physical activity when in upright posture. Units: g

mean*JerkStanding: Mean of the Jerk-based proxy for physical activity when in upright posture. Units: gs^{-1}

mean*DBALying: Mean of the DBA-based proxy for physical activity when in lying posture. Units: g

mean*JerkLying: Mean of the Jerk-based proxy for physical activity when in lying posture. Units: gs^{-1}

mean*DBALyingLeft: Mean of the DBA-based proxy for physical activity when in lying posture with lying side left. Units: g

mean*JerkLyingLeft: Mean of the Jerk-based proxy for physical activity when in lying posture with lying side left. Units: gs^{-1}

mean*DBALyingRight: Mean of the DBA-based proxy for physical activity when in lying posture with lying side right. Units: g

mean*JerkLyingRight: Mean of the Jerk-based proxy for physical activity when in lying posture with lying side right. Units: gs^{-1}

nBoutsStanding: Number of standing bouts (proportional if across intervals).

nBoutsLying: Number of lying bouts (proportional if across intervals).

nBoutsLyingLeft: Number of lying bouts with lying side left (proportional if across intervals).

nBoutsLyingRight: Number of lying bouts with lying side right (proportional if across intervals).

wMeanDurationStandingBout: Weighted-mean duration of standing bouts (weights are the proportions of the individual bouts overlapping with the respective intervals). Units: As specified via the `duration_units` argument.

wMeanDurationLyingBout: Weighted-mean duration of lying bouts (weights are the proportions of the individual bouts overlapping with the respective intervals). Units: As specified via the `duration_units` argument.

wMeanDurationLyingBoutLeft: Weighted-mean duration of lying bouts with lying side left (weights are the proportions of the individual bouts overlapping with the respective intervals). Units: As specified via the `duration_units` argument.

wMeanDurationLyingBoutRight: Weighted-mean duration of lying bouts with lying side right (weights are the proportions of the individual bouts overlapping with the respective intervals). Units: As specified via the `duration_units` argument.

`Triact` *R6 class for analyzing accelerometer data from cows*

Description

An object for containing and analyzing data from accelerometers attached to a hind leg of cows. Analyses focus on the lying behaviour and on the cow's level of physical activity as detailed in Simmler & Brouwers (2023).

Active bindings

`data` Raw accelerometer data and analysis results. Mainly modified by $load_... and the $add_... methods

Methods

Public methods:

- `Triact$new()`
- `Triact$load_files()`
- `Triact$load_table()`
- `Triact$check_orientation()`
- `Triact$add_lying()`
- `Triact$add_side()`
- `Triact$add_activity()`
- `Triact$summarize_intervals()`
- `Triact$summarize_bouts()`
- `Triact$extract_liedown()`
- `Triact$extract_standup()`

Method `new()`: Create a new triact object.

Usage:

```
Triact$new()
```

Returns: A new 'Triact' object.

Examples:

```
# create a Triact object
my_triact <- Triact$new()
```

Method `load_files()`: Import acceleration data files from one or several cows to the 'Triact' object. Importing multiple files from the same cows is possible but data should follow each other without any gaps in time (overlap is allowed as duplicates after concatenation will be removed). The filenames must allow unique identification of the cow (parameter: `id_substring`). Accelerometer sampling frequency must be consistent across the files. Acceleration should be in units of *g*. Triaxial, biaxial and uniaxial accelerometer data are allowed, but only triaxial data corresponding to relative body directions allows full functionality.

Important: Make sure to correctly specify how to map the axes as named by the accelerometer (e.g., x, y, z) to the body relative axes as used in triact (parameter: `timeFwdUpRight_cols`). See Simmler & Brouwers (2023) for an illustration.

Usage:

```
Triact$load_files(
        input,
        id_substring,
        timeFwdUpRight_cols = c(1, 2, 3, 4),
        time_format  = NULL,
        tz           = Sys.timezone(),
        start_time   = NULL,
        end_time     = NULL,
        sep          = "auto",
        skip         = "__auto__",
        parallel     = 1,
        ...
      )
```

Arguments:

`input` Specifies the input acceleration data files. Character vector with the name(s) of the file(s) or a directory containing the files (files can be in subdirectories). If it does not contain absolute paths, the directory or file name(s) are relative to the current working directory, `getwd()`.

`id_substring` Integer vector identifying the substring of the file names representing the unique identifier of the cows by character position: c(first, last), e.g, `c(1, 5)` for first to fifth character. Alternatively, a Perl-like regular expression matching the substring.

`timeFwdUpRight_cols` Integer vector specifying the columns containing the time, and the forward, up, and right axis acceleration data: c(time, fwd, up, right). Missing acceleration axes are specified as `NA`. A negative mathematical sign is used to indicate that the recorded data reflects the opposite direction (e.g., for backward acceleration, specify the forward acceleration column with a negative mathematical sign). *Default:* `c(1, 2, 3, 4)`

`time_format` Character vector specifying the date-time format corresponding to the acceleration files (syntax as in `strptime`). If `NULL` a date-time format as tried by as.POSIX* is expected. *Default:* `NULL`

`tz` Character vector specifying the time zone *Default:* `Sys.timezone()`

`start_time` Time from which on the data should be considered. Formatted as "%Y-%m-%d %H:%M:%OS" or in another format automatically tried by as.POSIX*. *Default:* `NULL`

`end_time` Time up to which the data should be considered. Formatted as "%Y-%m-%d %H:%M:%OS" or in another format automatically tried by as.POSIX*. *Default:* `NULL`

`sep` The separator between columns in the acceleration data files. If "auto", it is automatically detected by `data.table::fread`. *Default:* `"auto"`

`skip` An integer indicating the number of lines to skip before reading data from the files. Alternatively, a (sub)string indicating the line to start reading data, or "__auto__" for the automatic detection by `data.table::fread`.*Default:* `"__auto__"` `parallel` An integer indicating the number of files that are read in parallel. For `parallel` > 1 the reading of the individual file is set to single-threaded to avoid nested parallelization. This behavior can be overwritten by additionally passing `nThread` via `...` argument, which is passed on to `data.table::fread`. *Default:* `1`

`...` Further arguments passed to `data.table::fread`.

Examples:

```
# create a Triact object
my_triact <- Triact$new()

dir <- system.file("extdata", package = "triact")

my_triact$load_files(input = dir,
                     id_substring = c(1, 5),
                     timeFwdUpRight_cols = c(1, 2 ,3, 4),
                     skip = "*DATA")

# inspect imported data
head(my_triact$data)
```

Method `load_table()`**:** Import acceleration data from a data.frame-like table (see `cows_5hz` as an example). The table should contain the following columns:

colname	*type*	*description*
id	Factor	unique id for the cow
time	POSIXct	timestamp
acc_fwd	numeric	acceleration from *forward* axis (units: *g*)
acc_up	numeric	acceleration from *up* axis (units: *g*)
acc_right	numeric	acceleration from *right* axis (units: *g*)

The accelerometer sampling frequency must be the same across all cows (id). No time gaps are allowed (within data of one id). One or two of the acceleration columns may be missing, but the possible analyses are then limited.

Important: Make sure the accelerometer axes correctly represent body relative axes (forward, up, right). See Simmler & Brouwers (2023) for an illustration.

Usage:

```
Triact$load_table(table)
```

Usage (alternative syntax):

```
Triact$data <- table
```

Arguments:

`table` Data frame-like table containing the data to import. Must follow the requirements detailed in the description above.

Examples:

```
# create a Triact object
my_triact <- Triact$new()

my_triact$load_table(cows_5hz)

# inspect imported data
head(my_triact$data)
```

Method `check_orientation()`**:** Checks for each ID (unique identifier of the cow) whether the accelerometer may have been unintentionally mounted 180° rotated in the sagittal plane to the hind leg. If identified as such, the mathematical correction is applied, i.e. the forward and up axes are negated (multiplied by -1), i.e. the axes are mathematically rotated in order to comply with the orientation as specified when loading in the data. The check is `sum(acc_up > crit) < sum(acc_up < (-1 * crit))` with `crit = 0.5` by default.

Usage:

```
Triact$check_orientation(crit = 0.5, interactive = TRUE)
```

Arguments:

`crit` Critical value used in the check according to the expression noted in the description above. *Default:* `0.5`

`interactive` A logical value that indicates whether the function should interactively prompts the user before applying the correction. *Default:* `TRUE`

Examples:

```
# create a Triact object
my_triact <- Triact$new()

my_triact$load_table(cows_5hz)

my_triact$check_orientation()
```

Method `add_lying()`**:** Classify data into lying and standing and add results as column 'lying' to the 'Triact' object. Additionally, bouts (lying and standing) are uniquely numbered in column 'bout_id'. The simple rule-based algorithm is composed of three steps: In the first step, the *up* acceleration is filtered to obtain the gravity component of the signal. In the second step, a threshold is used to classify the filtered acceleration values into lying and standing. Finally, in the third step, lying bouts shorter than a given minimum duration are reclassified as standing. The last step can be performed analogous for standing bouts, but is not recommended by default. See Simmler & Brouwers (2023) for a detailed discussion.

Usage:

```
Triact$add_lying(
      filter_method = "median",
      crit_lie = 0.5,
      minimum_duration_lying = 30,
      minimum_duration_standing = NULL,
      add_filtered = FALSE,
      ...)
```

Arguments:

`filter_method` Filter method to be applied to obtain the gravity component of the acceleration on the *up* axis. Options are "median", for median filter, and "butter" for a bidirectional (zero-lag) Butterworth low-pass filter. Settings to the respective filter methods can be passed via the '...' argument (see below). *Default:* `"median"`

`crit_lie` Up-acceleration threshold for classifying into lying and standing. *Default:* `0.5`

`minimum_duration_lying` Minimum duration for lying bouts in seconds. Lying bouts shorter than this threshold are considered false and reclassified as standing. *Default:* `30`

`minimum_duration_standing` Minimum duration for standing bouts in seconds. Standing bouts shorter than this threshold are considered false and reclassified as lying. *Default:* `NULL`

`add_filtered` Logical value that indicates whether the filtered gravity component of the *up* acceleration should be added to the 'Triact' object. *Default:* `FALSE`

`...` Further arguments passed to the corresponding filter method as specified with the `filter_method` argument. For filter method 'median' the argument `window_size` specifies the window size in seconds (*default:* `10`). For filter method 'butter' the argument `cutoff` specifies the cutoff frequency for low-pass filtering in Hz (*default:* `0.1`) and the argument `order` specifies the Butterworth filter order (*default:* `1`).

Examples:

```
# create a Triact object
my_triact <- Triact$new()

my_triact$load_table(cows_5hz)

my_triact$add_lying()

# inspect result
head(my_triact$data)
```

Method `add_side()`**:** Classify lying bouts into left and right lying side and add results as column 'side' to the 'Triact' object. For each lying bout, the median acceleration over the entire duration is calculated and compared to a threshold to classify the bout as 'L' (left) or 'R' (right) lying side. See Simmler & Brouwers (2023) for details.

Usage:

```
Triact$add_side(left_leg, crit_left = if(left_leg) -0.5 else 0.5)
```

Arguments:

`left_leg` Logical indicating whether the accelerometers were attached to the left hind leg (`TRUE`) or to the right hind leg (`FALSE`). This information is used to choose the default value for `crit_left`. It is ignored if `crit_left` is specified by the user.

`crit_left` Right-acceleration threshold for classifying lying on left versus right side. *Default:* `-0.5` if `left_leg` is `TRUE`, else `0.5`

Examples:

```
# create a Triact object
my_triact <- Triact$new()
```

```
my_triact$load_table(cows_5hz)

my_triact$add_lying()

my_triact$add_side(left_leg = TRUE)

# inspect result
head(my_triact$data)
```

Method `add_activity()`**:** Calculate proxies for the physical activity level. By default, the L2 norm of the dynamic body acceleration (DBA) vector is calculated. The corresponding L1 norm is optionally available. Also, the L1 and L2 norms of the jerk vector can be calculated. By default, all activity values during lying bouts are 'adjusted' to zero, i.e., periods when cows are lying are considered as 'inactive' by definition. See Simmler & Brouwers (2023) for details.

Usage:

```
Triact$add_activity(
      dynamic_measure = "dba",
      norm = "L2",
      adjust = TRUE,
      filter_method = "median",
      keep_dynamic_measure = FALSE,
      ...)
```

Arguments:

`dynamic_measure` Type of dynamic measure to base the activity proxy on. Options are `"dba"`, for dynamic body acceleration, and `"jerk"`, for the jerk vector. One or both can be provided, e.g., `"dba"` or `c("dba", "jerk")`. *Default:* `"dba"`

`norm` The type of norm to be calculated. Options are `"L1"` and `"L2"`. One or both can be provided, e.g., `"L1"` or `c("L1", "L2")`. *Default:* `"L2"`

`adjust` A logical value that indicates whether the proxies for physical activity should be 'adjusted' to 0 during lying bouts, i.e., whether cows should be considered as inactive by definition when lying. *Default:* `TRUE`

`filter_method` Filter method to be used to determine the gravity component subtracted from the raw acceleration to obtain the dynamic body acceleration. Options are "median", for median filter, and "butter" for a bidirectional (zero-lag) Butterworth low-pass filter. Settings to the respective filter methods can be passed via the '...' argument (see below). *Default:* `"median"`

`keep_dynamic_measure` A logical value that indicates whether the intermediate data, being the dynamic body acceleration vector and/or the jerk vector, should be added to the 'Triact' object. *Default:* `FALSE`

`...` Further arguments passed to the corresponding filter method as specified with the `filter_method` argument. For filter method 'median' the argument `window_size` specifies the window size in seconds (*default:* `10`). For filter method 'butter' the argument `cutoff` specifies the cut-off frequency for low-pass filtering in Hz (*default:* `0.1`) and argument `order` specifies the Butterworth filter order (*default:* `1`).

Examples:

```
# create a Triact object
my_triact <- Triact$new()

my_triact$load_table(cows_5hz)

my_triact$add_lying()

my_triact$add_activity()

# inspect result
head(my_triact$data)
```

Method `summarize_intervals()`**:** Summarizes the data in the 'Triact' object, activity and lying behaviour, by regular intervals. The information contained in the output table depends on the analyses you added to the 'Triact' object using the $add_activity(), $add_lying(), and $add_side() methods. With `bout = TRUE` information on bouts per interval (number and mean bout duration) will be returned additionally. With `side = TRUE` summarized information such lying duration is additionally provided separately for the lying side (left/right). For measures such as the number of lying bouts or mean lying bout duration, a weighted mean ('wMean...') is calculated with the weights being the proportion of the individual bout overlapping with the respective interval. See Simmler & Brouwers (2023) for details.

Usage:

```
Triact$summarize_intervals(
      interval = "hour",
      lag_in_s = 0,
      duration_units = "mins",
      bouts = FALSE,
      side = FALSE,
      calc_for_incomplete = FALSE)
```

Arguments:

`interval` Character string specifying the intervals to be analyzed. Any unique English abbreviation valid for the 'unit' argument of `lubridate::floor_date` is allowed, e.g., `"hour"`, `"min"`, `"10 mins"`, and `"0.5 hours"`. *Default:* `"hour"`

`lag_in_s` Lag in seconds with respect to the full hour or full day. *Default:* `0`

`duration_units` Unit in which durations should be returned. Options are `"secs"`, `"mins"` and `"hours"`. *Default:* `"mins"`

`bouts` Logical indicating whether information on bouts should be additionally summarized. *Default:* `FALSE`

`side` Logical indicating whether lying side should be considered in the summary. *Default:* `FALSE`

`calc_for_incomplete` Logical indicating whether a complete summary should also be returned for the incompletely observed intervals (first and last interval for each cow) and for any parameter using information of incompletely observed bouts (first and last bout for each cow). Please note that these are ill-defined. *Default:* `FALSE`

Returns: A table with summaries by interval (rows). See interval_summary for a complete list.

Examples:

```
# create a Triact object
my_triact <- Triact$new()

my_triact$load_table(cows_5hz)

my_triact$add_lying()

int_summary <- my_triact$summarize_intervals()

# inspect result
head(int_summary)
```

Method `summarize_bouts()`**:** Summarizes the data in the 'Triact' object, activity and lying behaviour, by lying/standing bouts. The information contained in the output table depends on the analyses you added to the 'Triact' object using the $add_activity(), $add_lying(), and $add_side() methods.

Usage:

```
Triact$summarize_bouts(
      bout_type = "both",
      duration_units = "mins",
      calc_for_incomplete = FALSE)
```

Arguments:

`bout_type` Type of bout to be considered. Options are `"both"`, `"lying"`, and `"standing"`. *Default:* `"both"`

`duration_units` Units in which durations should be returned. Options are `"secs"`, `"mins"`, and `"hours"`. *Default:* `"mins"`

`calc_for_incomplete` Logical indicating whether a complete summary should also be returned for the incompletely observed bouts (first and last bout for each cow). Please note that these are ill-defined. *Default:* `FALSE`

Returns: A table with summaries by bout (rows). See bout_summary for a complete list.

Examples:

```
# create a Triact object
my_triact <- Triact$new()

my_triact$load_table(cows_5hz)

my_triact$add_lying()

bouts_summary <- my_triact$summarize_bouts()

# inspect result
head(bouts_summary)
```

Method `extract_liedown()`**:** Extracts data associated with liedown events (standing-to-lying transitions). Operates in two modes, see *Returns* section.

Usage:

```
Triact$extract_liedown(sec_before = 0, sec_after = 0)
```

Arguments:

`sec_before` From how many seconds before the liedown events data should be considered. *Default:* `0`

`sec_after` Until how many seconds after the liedown events data should be considered. *Default:* `0`

Returns: With default settings, a table with one entry per liedown event, with timestamp and bout_id of the lying bout, plus lying side information (if available). With parameters `sec_before` and/or `sec_after` > 0, a list containing individual tables per liedown event. These tables are extracts of all data in the 'Triact' object from within the defined time window around the liedown events.

Examples:

```
# create a Triact object
my_triact <- Triact$new()

my_triact$load_table(cows_5hz)

my_triact$add_lying()

l_downs <- my_triact$extract_liedown()

# inspect result
print(l_downs)
```

Method `extract_standup()`**:** Extracts data associated with standup events (lying-to-standing transitions). Operates in two modes, see *Returns* section.

Usage:

```
Triact$extract_standup(sec_before = 0, sec_after = 0)
```

Arguments:

`sec_before` From how many seconds before the standup events data should be considered. *Default:* `0`

`sec_after` Until how many seconds after the standup events data should be considered. *Default:* `0`

Returns: With default settings, a table with one entry per standup event, with timestamp and bout_id of the lying bout, plus lying side information (if available). With parameters `sec_before` and/or `sec_after` > 0, a list containing individual tables per standup event. These tables are extracts of all data in the 'Triact' object from within the defined time window around the standup events.

Examples:

```
# create a Triact object
my_triact <- Triact$new()

my_triact$load_table(cows_5hz)
```

```
my_triact$add_lying()

st_ups <- my_triact$extract_standup()

# inspect result
print(st_ups)
```

References

Simmler. M., Brouwers S., 2023. Triact package for R: Analyzing the lying behavior of cows from accelerometer data. pdf

Examples

```
## Please read the "introduction" vignette for more detailed examples

## -------------------------
# method '$new'
## -------------------------

# create a Triact object
my_triact <- Triact$new()

## -------------------------
# method '$load_files'
## -------------------------


# create a Triact object
my_triact <- Triact$new()

dir <- system.file("extdata", package = "triact")

my_triact$load_files(input = dir,
                     id_substring = c(1, 5),
                     timeFwdUpRight_cols = c(1, 2 ,3, 4),
                     skip = "*DATA")

# inspect imported data
head(my_triact$data)


## -------------------------
# method 'load_table'
## -------------------------

# create a Triact object
my_triact <- Triact$new()
```

```
my_triact$load_table(cows_5hz)

# inspect imported data
head(my_triact$data)

## -------------------------
# method 'check_orientation'
## -------------------------

my_triact$check_orientation()

## -------------------------
# method 'add_lying'
## -------------------------

my_triact$add_lying()

# inspect result
head(my_triact$data)

## -------------------------
# method 'add_side'
## -------------------------

my_triact$add_side(left_leg = TRUE)

# inspect result
head(my_triact$data)

## -------------------------
# method 'add_activity'
## -------------------------

my_triact$add_activity()

# inspect result
head(my_triact$data)

## -------------------------
# method 'summarize_intervals'
## -------------------------

int_summary <- my_triact$summarize_intervals()

# inspect result
head(int_summary)

## -------------------------
# method 'summarize_bouts'
## -------------------------

bouts_summary <- my_triact$summarize_bouts()
```

```
# inspect result
head(bouts_summary)

## -------------------------
# method 'extract_liedown'
## -------------------------

l_downs <- my_triact$extract_liedown()

# inspect result
print(l_downs)

## -------------------------
# method 'extract_standup'
## -------------------------

st_ups <- my_triact$extract_standup()

# inspect result
print(st_ups)
```

Index