Package ‘ggetho’

December 8, 2017

Title Visualise High-Throughput Behavioural (i.e. Ethomics) Data

Date 2017-07-25

Version 0.3.0.9003

Description Uses ggplot to represent animal behaviour data, generally recorded over multiple days.

Depends R (>= 3.00),
ggplot2,
behavr

Imports data.table,
hms,
stringr,
scales,
labeling

Suggests testthat,
covr,
knitr

License GPL-3

Encoding UTF-8

LazyData true

URL `https://github.com/rethomics/ggetho`

BugReports `https://github.com/rethomics/ggetho/issues`

RoxygenNote 6.0.1

Roxygen list(markdown = TRUE)

R topics documented:

---

ggetho *Prepare a ggplot object to represent behavioural data*

Description

This function summarises a variable of interest (y or z axis) in order to subsequently represent it over time (x axis) (either using `ggplot2` or the of plotting functions provided in ‘ggetho“).

Usage

```
ggetho(data, mapping, summary_FUN = mean, summary_time_window = mins(30),
  time_wrap = NULL, time_offset = 0, multiplot = NULL,
  multiplot_period = hours(24), ...)
```

Arguments

`data`	behavr table containing the data and metadata
`mapping`	default list of aesthetic mappings to use for plot
`summary_FUN`	method (function) used to summarise `variable` over time (typically, the mean)
`summary_time_window`	width (in seconds) of the time window to compute a summary on
`time_wrap`	time (in seconds) used to wrap the data (see details)
`time_offset`	time offset (i.e. phase, in seconds) when using `time_wrap`
`multiplot`	integer, greater than two, or NULL, the default (see details)
`multiplot_period`	the duration of the period when mutiplotting (see details)
`...`	additional arguments to be passed to `ggplot2::ggplot()`

Details

`time_wrap` is typically used to express time relatively to the start of the the day. In other words, it can help be used to pull all days together in one representative day. In this case, `time_wrap = hours(24)`. Instead of representing data from the start of the day, it can be done from any offset, using `time_offset`. For instance, `time_offset = hours(12)` puts the circadian reference (ZT0) in the middle of the plot.

Multiplots is a generalistion of double-plotting, tripple-plotting... This type or representation is usefull to understand periodic behaviours. When `multiplot` is not NULL, data is repeated as many time along the x axis to generate a double (when `multiplot=2`) plotted actogram The y axis then is the period (typically the day) onset. It is possible to set duration of the period, which is typically 24h to arbitrary values using the `multiplot_period` argument.

Value

an initial plot object that can be further edited.

See Also

- stat_pop_etho to show population trend by aggregating individuals over time
- stat_tile_etho to show variable of interest as colour intensity
- stat_ld_annotations to show light and dark phases on the plot @references
- The relevant rethomic tutorial section

Examples

```
# We start by making a to dataset with 20 animals
metadata <- data.table(id= sprintf("toy_experiment|%02d", 1:20),
                   condition=c("A","B"))
dt <- toy_activity_data(metadata, 3)
# We build a plot object with **nothing inside** (just the axis)
# we want to show proportion of time sleeping  on the y axis:
pl <- ggetho(dt, aes(y=asleep))
pl
# Sometimes, the variable of interest in not on the y axis, but on z axis (colour scale).
# When we do not provide a y axis,
# ggetho will make a ID fo each animal and display them on separate rows
pl <- ggetho(dt, aes(z=asleep))
pl
# this one is the same type, but groups the animals by condition
pl <- ggetho(dt, aes(z=asleep,y=condition))
pl
# sorting with paste
pl <- ggetho(dt, aes(z=asleep,y=paste(condition, id)))
pl

# we want to summarise (wrap) data along a circadian day:
pl <- ggetho(dt, aes(y=asleep), time_wrap=hours(24))
pl

# double ploted actogram:
pl <- ggetho(dt,
              aes(z=moving),
              multiplot = 2,
              multiplot_period = hours(24))
pl
# then use `+ stat_tile_etho()` , or `+ stat_bar_tile_etho()`
```

`id_labeller` *A facet labeller for* `id`

Description

This function returns a ggplot2::labeller that displays the `id` on sevreal lines to improve readability.

Usage

```
id_labeller(labels)
```

Arguments

`labels`	Data frame of labels. Usually contains only one element, but facetting over multiple factors entails multiple label variables.

See Also

ggplot2::labeller, to make your own labellers

Examples

```
library(behavr)
metadata <- data.frame(
     id = sprintf("2017-09-01 20:00:12|toy_experiment_a_very_long_name|%02d",1:20),
     condition=c("A","B"))
dt <- toy_activity_data(metadata, duration=hours(2))
pl <- ggetho(dt, aes(y=asleep)) + stat_pop_etho()
## Without labelling
pl + facet_wrap( ~ id)

## With labeller
pl + facet_wrap( ~ id, labeller = id_labeller)
```

`stat_bar_tile_etho` *Display a behavioural variable of interest as colour intensity value or bar height*

Description

These function shows the temporal trend (time on the x axis) of a varible of interest (z axis) as either colour instensity (`stat_tile_etho`) or using the hight of the tiles (`stat_bar_tile_etho`). In both cases, the y axis is a discrete variable such as a treatment or the id of animals.

Usage

```
stat_bar_tile_etho(mapping = NULL, data = NULL, geom = "bar_tile",
  position = "identity", ..., method = mean, method.args = list(),
  na.rm = FALSE, show.legend = NA, inherit.aes = TRUE)

stat_tile_etho(mapping = NULL, data = NULL, geom = "raster",
  position = "identity", ..., method = mean, method.args = list(),
  na.rm = FALSE, show.legend = NA, inherit.aes = TRUE)
```

Arguments

`mapping`	Set of aesthetic mappings created by `aes` or `aes_`. If specified and `inherit.aes = TRUE` (the default), it is combined with the default mapping at the top level of the plot. You must supply `mapping` if there is no plot mapping.
`data`	The data to be displayed in this layer. There are three options: If `NULL`, the default, the data is inherited from the plot data as specified in the call to `ggplot`. A `data.frame`, or other object, will override the plot data. All objects will be fortified to produce a data frame. See `fortify` for which variables will be created. A `function` will be called with a single argument, the plot data. The return value must be a `data.frame.`, and will be used as the layer data.
`geom`	The geometric object to use display the data
`position`	Position adjustment, either as a string, or the result of a call to a position adjustment function.
`...`	other arguments passed on to `layer`. These are often aesthetics, used to set an aesthetic to a fixed value, like `color = "red"` or `size = 3`. They may also be parameters to the paired geom/stat.
`method`	function used to compute the aggregate, when grouping individuals on the same row. The default is mean. median, min, max are other examples of other functions one can use.
`method.args`	List of additional arguments passed on to the modelling function defined by `method`.
`na.rm`	If `FALSE`, the default, missing values are removed with a warning. If `TRUE`, missing values are silently removed.
`show.legend`	logical. Should this layer be included in the legends? `NA`, the default, includes if any aesthetics are mapped. `FALSE` never includes, and `TRUE` always includes.
`inherit.aes`	If `FALSE`, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn't inherit behaviour from the default plot specification, e.g. `borders`.

References

- The relevant rethomic tutorial section

See Also

- ggetho to generate a plot object
- stat_pop_etho to show population trend by aggregating individuals over time
- stat_ld_annotations to show light and dark phases on the plot

Other layers: `stat_ld_annotations`, `stat_pop_etho`

Examples

```
# we start by making a to dataset with 20 animals
metadata <- data.frame(id = sprintf("toy_experiment | %02d", 1:20),
                   age=c(1, 5, 10, 20),
                   condition=c("A","B"))
print(metadata)
dt <- toy_activity_data(metadata,3)
# We build a plot object
pl <-  ggetho(dt, aes(z=asleep))
# A standard plot one row per animal:
pl + stat_tile_etho()
# We can also group animals per condition and calculate the average sleep
pl <-  ggetho(dt, aes(z=asleep, y=condition))
pl + stat_tile_etho()

# We can sort by adding condition AND id on the y axis:
pl <-  ggetho(dt, aes(z=asleep, y= interaction(id, condition)))
pl + stat_tile_etho()
# Same if we want to sort by age
pl <-  ggetho(dt, aes(z=asleep, y= interaction(id, age)))
pl + stat_tile_etho()

# Instead, of the average, maybe we want to show the highest (max)
# posible value of sleep for any time point
pl + stat_tile_etho(method=max)
# we can also use stat_bar_tile as an alternative
pl + stat_bar_tile_etho()
```

---

stat_ld_annotations *Compute and display light/dark annotations onto a plot object*

Description

This function is used to show light and dark (L and D) phases as boxes on top a plot.

Usage

```
stat_ld_annotations(mapping = NULL, data = NULL, position = "identity",
  ld_colours = c("white", "black"), ypos = "bottom", height = 0.03,
  period = hours(24), phase = 0, outline = "black", x_limits = NA, ...,
  na.rm = FALSE, show.legend = FALSE, inherit.aes = TRUE)
```

Arguments

`mapping`	Set of aesthetic mappings created by `aes` or `aes_`. If specified and `inherit.aes = TRUE` (the default), it is combined with the default mapping at the top level of the plot. You must supply `mapping` if there is no plot mapping.

`data`	The data to be displayed in this layer. There are three options: If `NULL`, the default, the data is inherited from the plot data as specified in the call to `ggplot`. A `data.frame`, or other object, will override the plot data. All objects will be fortified to produce a data frame. See `fortify` for which variables will be created. A `function` will be called with a single argument, the plot data. The return value must be a `data.frame.`, and will be used as the layer data.
`position`	Position adjustment, either as a string, or the result of a call to a position adjustment function.
`ld_colours`	character vector of length 2 naming the colours for light and dark phases, respectively. The default is white and black.
`ypos`	position and height of the annotation on the y axis. It can be either `"top"` of `"bottom"`. The default, `"bottom"` will put the labels below any data.
`height`	relative height of the rectangles. The default is 3 percent (0.03).
`period, phase`	period and phase (in seconds) of the LD cycle.
`outline`	colour of the border of the rectangles. `NA` means no border.
`x_limits`	numerical vector of length 2 for the start and end of the annotations (in seconds). The default, `NA`, uses the full range of the plotted data.
`...`	other arguments passed on to `layer`. These are often aesthetics, used to set an aesthetic to a fixed value, like `color = "red"` or `size = 3`. They may also be parameters to the paired geom/stat.
`na.rm`	If `FALSE`, the default, missing values are removed with a warning. If `TRUE`, missing values are silently removed.
`show.legend`	logical. Should this layer be included in the legends? `NA`, the default, includes if any aesthetics are mapped. `FALSE` never includes, and `TRUE` always includes.
`inherit.aes`	If `FALSE`, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn't inherit behaviour from the default plot specification, e.g. `borders`.

References

- The relevant rethomic tutorial section

See Also

- ggetho to generate a plot object

Other layers: `stat_bar_tile_etho`, `stat_pop_etho`

Examples

```
library(behavr)
# we start by making a to dataset with 20 animals
metadata <- data.frame(id = sprintf("toy_experiment | %02d", 1:20),
                       condition=c("A","B"))
```

```
dt <- toy_activity_data(metadata,3)
# We build a plot object
pl <-  ggetho(dt, aes(y=asleep)) + stat_pop_etho()
pl + stat_ld_annotations()
# We can also put the annotations in the background:
pl <-  ggetho(dt, aes(y=asleep)) +
                 stat_ld_annotations(outline=NA) +
                 stat_pop_etho()
pl
# different colours (e.g. DD)
pl + stat_ld_annotations(ld_colour=c("grey", "black"))
# shorter period
pl + stat_ld_annotations(period=hours(22), phase=hours(3))
# on a tile plot:
pl <-  ggetho(dt, aes(z=asleep)) + stat_tile_etho()
pl + stat_ld_annotations()
```

---

stat_pop_etho — *Compute and display a population aggregate for a behavioural variable of interest*

Description

This function displays the temporal (time on the x axis) trend of variable of interest, on the y axis as a line with error bars.

Usage

```
stat_pop_etho(mapping = NULL, data = NULL, geom = "smooth",
  position = "identity", ..., method = mean_se, method.args = list(),
  show.legend = NA, inherit.aes = TRUE)
```

Arguments

`mapping`	Set of aesthetic mappings created by `aes` or `aes_`. If specified and `inherit.aes = TRUE` (the default), it is combined with the default mapping at the top level of the plot. You must supply `mapping` if there is no plot mapping.
`data`	The data to be displayed in this layer. There are three options: If `NULL`, the default, the data is inherited from the plot data as specified in the call to `ggplot`. A `data.frame`, or other object, will override the plot data. All objects will be fortified to produce a data frame. See `fortify` for which variables will be created. A `function` will be called with a single argument, the plot data. The return value must be a `data.frame.`, and will be used as the layer data.
`geom`	The geometric object to use display the data

`position`	Position adjustment, either as a string, or the result of a call to a position adjustment function.
`...`	other arguments passed on to `layer`. These are often aesthetics, used to set an aesthetic to a fixed value, like `color = "red"` or `size = 3`. They may also be parameters to the paired geom/stat.
`method`	function used to compute the aggregate and error bars. It should return (`y`, `ymin` and `ymax`). The default is ggplot2::mean_se, which computes the mean + or - standard error. ggplot2::mean_cl_boot can be used instead to generate bootstrap confidence interval.
`method.args`	List of additional arguments passed on to the modelling function defined by `method`.
`show.legend`	logical. Should this layer be included in the legends? `NA`, the default, includes if any aesthetics are mapped. `FALSE` never includes, and `TRUE` always includes.
`inherit.aes`	If `FALSE`, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn't inherit behaviour from the default plot specification, e.g. `borders`.

References

- The relevant rethomic tutorial section

See Also

- ggetho to generate a plot object
- stat_tile_etho to show variable of interest as colour intensity
- stat_ld_annotations to show light and dark phases on the plot
- ggplot2::stat_smooth to understand how to change the type of error bars etc

Other layers: `stat_bar_tile_etho`, `stat_ld_annotations`

Examples

```
library(behavr)
metadata <- data.frame(id = sprintf("toy_experiment | %02d", 1:20),
                   age=c(1, 5, 10, 20),
                   condition=c("A","B"))
dt <- toy_activity_data(metadata,3)
# We build a plot object
pl <-  ggetho(dt, aes(y=asleep))
# A standard plot of the whole population:
pl + stat_pop_etho()
# We can also split by condition, and display the two population on different facets:
pl + stat_pop_etho() + facet_grid(condition ~ .)

# Instead, we can use different colour for separate conditions:
pl <-  ggetho(dt, aes(y=asleep, colour=condition))
pl + stat_pop_etho()

#sometimes, we also have numeric condition (e.g. age)
```

```
pl <-  ggetho(dt, aes(y=asleep, colour=age))
pl + stat_pop_etho()
# sometimes we want to aggreate several days of data to one circadian day (i.e. time wrapping)
# here, we also plot the invert of moving (!moving)
pl <-  ggetho(dt, aes(y=!moving), time_wrap=hours(24))
pl + stat_pop_etho()
```

`time_scales` *Scales for durations*

Description

Scales used to represent behaviour durations

Usage

```
scale_x_days(name = "Time (day)", breaks = waiver(),
  minor_breaks = waiver(), labels = waiver(), limits = NULL,
  expand = waiver(), oob = scales::censor, na.value = NA_real_,
  position = "bottom", time_wrap = NULL)

scale_x_hours(name = "Time (h)", breaks = waiver(),
  minor_breaks = waiver(), labels = waiver(), limits = NULL,
  expand = waiver(), oob = scales::censor, na.value = NA_real_,
  position = "bottom", time_wrap = NULL)

scale_x_seconds(name = "Time (s)", breaks = waiver(),
  minor_breaks = waiver(), labels = waiver(), limits = NULL,
  expand = waiver(), oob = scales::censor, na.value = NA_real_,
  position = "bottom", time_wrap = NULL)
```

Arguments

`name` The name of the scale. Used as axis or legend title. If `NULL`, the default, the name of the scale is taken from the first mapping used for that aesthetic.

`breaks` One of:

- `NULL` for no breaks
- `waiver()` for the default breaks computed by the transformation object
- A numeric vector of positions
- A function that takes the limits as input and returns breaks as output

`minor_breaks` One of:

- `NULL` for no minor breaks
- `waiver()` for the default breaks (one minor break between each major break)
- A numeric vector of positions

	• A function that given the limits returns a vector of minor breaks.
`labels`	One of: • `NULL` for no labels • `waiver()` for the default labels computed by the transformation object • A character vector giving labels (must be same length as `breaks`) • A function that takes the breaks as input and returns labels as output
`limits`	A numeric vector of length two providing limits of the scale. Use `NA` to refer to the existing minimum or maximum.
`expand`	A numeric vector of length two giving multiplicative and additive expansion constants. These constants ensure that the data is placed some distance away from the axes. The defaults are `c(0.05, 0)` for continuous variables, and `c(0, 0.6)` for discrete variables.
`oob`	Function that handles limits outside of the scale limits (out of bounds). The default replaces out of bounds values with NA.
`na.value`	Missing values will be replaced with this value.
`position`	The position of the axis. "left" or "right" for vertical scales, "top" or "bottom" for horizontal scales
`time_wrap`	duration (in seconds) used to wrap the lanbels of the time axis.

Details

`time_wrap` is useful when for instance wanting to express time within a day (ZT).

References

- The relevant rethomic tutorial section

See Also

- ggetho to generate a plot object
- ggplot2::scale_x_continuous, the defaut ggplot scale, to understand limits, breaks, labels and name

Examples

```
# we generate some data
metadata <- data.frame(id = sprintf("toy_experiment | %02d", 1:20),
                   condition=c("A","B"))
dt <- toy_activity_data(metadata,3)
# then, a simple plot
pl <-  ggetho(dt, aes(y=asleep)) + stat_pop_etho()
pl + scale_x_hours(breaks = days(c(1, 2)))
pl + scale_x_hours()
pl + scale_x_days(breaks = days(c(1, 2)))
 pl + scale_x_days()
# on a shorter time scale
pl <-  ggetho(dt[t < hours(5)], aes(z=asleep)) + stat_tile_etho()
pl + scale_x_hours()
pl + scale_x_hours(breaks = hours(1:4))
pl + scale_x_seconds(breaks = hours(1:4))

# time wraping
pl <-  ggetho(dt[t < days(2)], aes(y=asleep)) + stat_pop_etho()
pl + scale_x_hours(time_wrap = hours(24))
```

Index