

# Package ‘ggetho’

October 5, 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:

boot_ci . . . . .	2
ggetho . . . . .	2
id_labeller . . . . .	4
scale_x_days . . . . .	4
stat_ld_annotatons . . . . .	6
stat_pop_etho . . . . .	8
stat_tile_etho . . . . .	9

**Index****12**


---

boot_ci	<i>Bootstrap confidence interval</i>
---------	--------------------------------------

---

**Description**

Compute the mean of a variable, and the quantiles after bootstrap resampling.

**Usage**

```
boot_ci(y, r = 5000, ci = 0.95)
```

**Arguments**

y	Numeric vector
r	Number of replicates to draw.
ci	Confidence interval to draw from the empirical distribution.

**See Also**

boot\_ci is intended to be used as the method argument of [stat\\_pop\\_etho](#). Other functions, such as [ggplot2::mean\\_se](#), can be used to generate error bars.

---

ggetho	<i>Prepare a ggplot object to represent behavioural data</i>
--------	--

---

**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 = NULL, ...)
```

**Arguments**

data	<a href="#">behavr</a> 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
...	additional arguments to be passed to <a href="#">ggplot2::ggplot()</a>

## 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.

## Value

an initial plot object that can be further edited.

## Author(s)

Quentin Geissmann (<qgeissmann@gmail.com>)

## 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

## 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:
p1 <- ggetho(dt, aes(y=asleep))
p1
# Sometimes, the variable of interest is not on the y axis, but on z axis (colour scale).
# When we do not provide a y axis,
# ggetho will make a ID for each animal and display them on separate rows
p1 <- ggetho(dt, aes(z=asleep))
p1
# this one is the same type, but groups the animals by condition
p1 <- ggetho(dt, aes(z=asleep,y=condition))
p1
# sorting with paste
p1 <- ggetho(dt, aes(z=asleep,y=paste(condition, id)))
p1

# we want to summarise (wrap) data along a circadian day:
p1 <- ggetho(dt, aes(y=asleep), time_wrap=hours(24))
p1
```

---

id_labeller	<i>A facet labeller for id</i>
-------------	--------------------------------

---

### Description

This function returns a `ggplot2::labeller` that displays the id on several 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))
p1 <- ggeth(dt, aes(y=asleep)) + stat_pop_etho()
## Without labelling
p1 + facet_wrap( ~ id)

## With labeller
p1 + facet_wrap( ~ id, labeller = id_labeller)
```

---

scale_x_days	<i>Scales for durations</i>
--------------	-----------------------------

---

### 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")
```

```
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")
```

```
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")
```

**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: <ul style="list-style-type: none"> <li>• NULL for no breaks</li> <li>• <code>waiver()</code> for the default breaks computed by the transformation object</li> <li>• A numeric vector of positions</li> <li>• A function that takes the limits as input and returns breaks as output</li> </ul>
minor_breaks	One of: <ul style="list-style-type: none"> <li>• NULL for no minor breaks</li> <li>• <code>waiver()</code> for the default breaks (one minor break between each major break)</li> <li>• A numeric vector of positions</li> <li>• A function that given the limits returns a vector of minor breaks.</li> </ul>
labels	One of: <ul style="list-style-type: none"> <li>• NULL for no labels</li> <li>• <code>waiver()</code> for the default labels computed by the transformation object</li> <li>• A character vector giving labels (must be same length as breaks)</li> <li>• A function that takes the breaks as input and returns labels as output</li> </ul>
limits	A numeric vector of length two providing limits of the scale. Use NA to refer to the existing minimum or maximum.
expand	Vector of range expansion constants used to add some padding around the data, to ensure that they are placed some distance away from the axes. Use the convenience function <a href="#">expand_scale()</a> to generate the values for the expand argument. The defaults are to expand the scale by 5% on each side for continuous variables, and by 0.6 units on each side 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

### See Also

- [ggplot2::scale\\_x\\_time](#) – to print the time as HH:MM:SS

### 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()
```

---

stat_ld_annotations	<i>Compute and display light/dark annotations onto a plot object</i>
---------------------	--

---

### 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 = "auto", height = "auto",
  period = hours(24), phase = 0, ..., na.rm = FALSE,
  show.legend = FALSE, inherit.aes = TRUE)
```

### Arguments

mapping	Set of aesthetic mappings created by <a href="#">aes()</a> or <a href="#">aes_()</a> . If specified and <code>inherit.aes = TRUE</code> (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 <code>NULL</code> , the default, the data is inherited from the plot data as specified in the call to <a href="#">ggplot()</a> . A <code>data.frame</code> , or other object, will override the plot data. All objects will be fortified to produce a data frame. See <a href="#">fortify()</a> for which variables will be created. A function will be called with a single argument, the plot data. The return value must be a <code>data.frame.</code> , 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, height	The position and height of the annotation on the y axis. The defaults, "auto" will put the labels below any data.
period, phase	period and phase (in seconds) of the LD cycle.
...	other arguments passed on to <a href="#">layer()</a> . These are often aesthetics, used to set an aesthetic to a fixed value, like <code>color = "red"</code> or <code>size = 3</code> . 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. It can also be a named logical vector to finely select the aesthetics to display.
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. <a href="#">borders()</a> .

### See Also

Useful links:

- [ggetho](#) to generate a plot object
- TODO Tutorial for this function

Other layers: [stat\\_pop\\_etho](#), [stat\\_tile\\_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_annotatations()
# different colours (e.g. DD)
pl + stat_ld_annotatations(ld_colour=c("grey", "black"))
# shorter period
pl + stat_ld_annotatations(period=hours(22), phase=hours(3))
# on a tile plot:
pl <- ggetho(dt, aes(z=asleep)) + stat_tile_etho()
pl + stat_ld_annotatations()
```

---

stat_pop_etho	<i>Compute and display a population aggregate for a behavioural variable of interest</i>
---------------	--

---

## 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 <a href="#">aes()</a> or <a href="#">aes_()</a> . If specified and <code>inherit.aes = TRUE</code> (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 <code>NULL</code> , the default, the data is inherited from the plot data as specified in the call to <a href="#">ggplot()</a> . A <code>data.frame</code> , or other object, will override the plot data. All objects will be fortified to produce a data frame. See <a href="#">fortify()</a> for which variables will be created. A function will be called with a single argument, the plot data. The return value must be a <code>data.frame</code> , 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 <a href="#">layer()</a> . These are often aesthetics, used to set an aesthetic to a fixed value, like <code>color = "red"</code> or <code>size = 3</code> . 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 <a href="#">ggplot2::mean_se</a> , which computes the mean + or - standard error. <a href="#">boot_ci</a> can be used instead to generate bootstrap confidence interval.
method.args	List of additional arguments passed on to the modelling function defined by <code>method</code> .
show.legend	logical. Should this layer be included in the legends? <code>NA</code> , the default, includes if any aesthetics are mapped. <code>FALSE</code> never includes, and <code>TRUE</code> always includes. It can also be a named logical vector to finely select the aesthetics to display.
inherit.aes	If <code>FALSE</code> , 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. <a href="#">borders()</a> .



**See Also**

- [ggetho](#) to generate a plot object
- [stat\\_tile\\_etho](#) to show variable of interest as colour intensity
- [stat\\_ld\\_annotatons](#) to show light and dark phases on the plot
- TODO Tutorial for this function

Other layers: [stat\\_ld\\_annotatons](#), [stat\\_tile\\_etho](#)

**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
p1 <- ggetho(dt, aes(y=asleep))
# A standard plot of the whole population:
p1 + stat_pop_etho()
# We can also split by condition, and display the two population on different facets:
p1 + stat_pop_etho() + facet_grid(condition ~ .)

# Instead, we can use different colour for separate conditions:
p1 <- ggetho(dt, aes(y=asleep, colour=condition))
p1 + stat_pop_etho()

#sometimes, we also have numeric condition (e.g. age)
p1 <- ggetho(dt, aes(y=asleep, colour=age))
p1 + stat_pop_etho()
# sometimes we want to aggregate several days of data to one circadian day (i.e. time wrapping)
# here, we also plot the invert of moving (!moving)
p1 <- ggetho(dt, aes(y=!moving), time_wrap=hours(24))
p1 + stat_pop_etho()
```

---

stat\_tile\_etho

---

*Display a behavioural variable of interest as colour intensity value*


---

**Description**

This function shows the temporal trend (time on the x axis) of a variable of interest as colour intensity (z axis). The y axis is a discrete variable such as a treatment or the id of animals.

**Usage**

```
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 <a href="#">aes()</a> or <a href="#">aes_()</a> . If specified and <code>inherit.aes = TRUE</code> (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	<p>The data to be displayed in this layer. There are three options:</p> <p>If <code>NULL</code>, the default, the data is inherited from the plot data as specified in the call to <a href="#">ggplot()</a>.</p> <p>A <code>data.frame</code>, or other object, will override the plot data. All objects will be fortified to produce a data frame. See <a href="#">fortify()</a> for which variables will be created.</p> <p>A function will be called with a single argument, the plot data. The return value must be a <code>data.frame</code>, and will be used as the layer data.</p>
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 <a href="#">layer()</a> . These are often aesthetics, used to set an aesthetic to a fixed value, like <code>color = "red"</code> or <code>size = 3</code> . 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 <a href="#">mean</a> . <a href="#">median</a> , <a href="#">min</a> , <a href="#">max</a> are other examples of other functions one can use.
method.args	List of additional arguments passed on to the modelling function defined by <code>method</code> .
na.rm	If <code>FALSE</code> , the default, missing values are removed with a warning. If <code>TRUE</code> , missing values are silently removed.
show.legend	logical. Should this layer be included in the legends? <code>NA</code> , the default, includes if any aesthetics are mapped. <code>FALSE</code> never includes, and <code>TRUE</code> always includes. It can also be a named logical vector to finely select the aesthetics to display.
inherit.aes	If <code>FALSE</code> , 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. <a href="#">borders()</a> .

**See Also**

- [ggetho](#) to generate a plot object
- [stat\\_pop\\_etho](#) to show population trend by aggregating individuals over time
- [stat\\_ld\\_annotatons](#) to show light and dark phases on the plot
- TODO Tutorial for this function <http://gilestrolab.github.io/rethomics/tutorial/todo>

Other layers: [stat\\_ld\\_annotatons](#), [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),
                       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)
# possible value of sleep for any time point
pl + stat_tile_etho(method=max)
```

# Index

`aes()`, [6](#), [8](#), [10](#)  
`aes_()`, [6](#), [8](#), [10](#)

`behavr`, [2](#)  
`boot_ci`, [2](#), [8](#)  
`borders()`, [7](#), [8](#), [10](#)

`expand_scale()`, [5](#)

`fortify()`, [6](#), [8](#), [10](#)

`ggetho`, [2](#), [7](#), [9](#), [10](#)  
`ggplot()`, [6](#), [8](#), [10](#)  
`ggplot2::ggplot()`, [2](#)  
`ggplot2::labeller`, [4](#)  
`ggplot2::mean_se`, [2](#), [8](#)  
`ggplot2::scale_x_time`, [6](#)

`id_labeller`, [4](#)

`layer()`, [7](#), [8](#), [10](#)

`max`, [10](#)  
`mean`, [10](#)  
`median`, [10](#)  
`min`, [10](#)

`scale_x_days`, [4](#)  
`scale_x_hours (scale_x_days)`, [4](#)  
`scale_x_seconds (scale_x_days)`, [4](#)  
`stat_ld_annotatons`, [3](#), [6](#), [9](#), [10](#)  
`stat_pop_etho`, [2](#), [3](#), [7](#), [8](#), [10](#)  
`stat_tile_etho`, [3](#), [7](#), [9](#), [9](#)