Package ‘gghalves’

May 30, 2022

Title Compose Half-Half Plots Using Your Favourite Geoms
Version 0.1.3
Description A ‘ggplot2’ extension for easy plotting of half-half geom combinations. Think half boxplot and half jitterplot, or half violinplot and half dotplot.
URL https://github.com/erocoar/gghalves
License MIT + file LICENSE
Encoding UTF-8
Depends ggplot2 (>= 3.0.0), R (>= 3.4.0)
Imports grid, gtable, grDevices
RoxygenNote 7.2.0
Suggests knitr, rmarkdown, dplyr
Collate 'utilities.R' 'geom_half_dotplot.R' 'geom_half_boxplot.R'
    'geom_half_point.R' 'geom_half_point_panel.R'
    'geom_half_violin.R' 'stat-half-bindot.R' 'stat-half-point.R'
    'stat-half-ydensity.R' 'ggproto-classes.R'
VignetteBuilder knitr
NeedsCompilation no
Author Frederik Tiedemann [aut, cre]
Maintainer Frederik Tiedemann <fj.tiedemann@googlemail.com>
Repository CRAN
Date/Publication 2022-05-30 10:10:08 UTC

R topics documented:

GeomHalfDotplot . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 2
geom_half_boxplot . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 2
geom_half_dotplot . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 4
geom_half_point . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 7
geom_half_point_panel . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 8
geom_half_violin . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 10

Index 13
GeomHalfDotplot  
gghalves extensions to ggplot2

Description

gghalves extensions to ggplot2

geom_half_boxplot  
A half boxplot

Description

A half boxplot

Usage

geom_half_boxplot(
  mapping = NULL,
  data = NULL,
  stat = "boxplot",
  position = "dodge2",
  ...
  side = "l",
  center = FALSE,
  nudge = 0,
  outlier.colour = NULL,
  outlier.color = NULL,
  outlier.fill = NULL,
  outlier.shape = 19,
  outlier.size = 1.5,
  outlier.stroke = 0.5,
  outlier.alpha = NULL,
  notch = FALSE,
  notchwidth = 0.5,
  varwidth = FALSE,
  errorbar.draw = TRUE,
  errorbar.length = 0.5,
  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. A function can be created from a `formula` (e.g. `~ head(.x, 10)`).

stat
Use to override the default connection between `geom_boxplot()` and `stat_boxplot()`.

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 `colour = "red"` or `size = 3`. They may also be parameters to the paired geom/stat.

side
The side of the half-geom, "l" for left and "r" for right, defaults to "l".

center
Boolean whether to center the half-boxplot instead of aligning it to its respective side.

nudge
Add space between the boxplot and the middle of the space allotted to a given factor on the x-axis.

outlier.colour, outlier.color, outlier.fill, outlier.shape, outlier.size, outlier.stroke, outlier.alpha
Default aesthetics for outliers. Set to `NULL` to inherit from the aesthetics used for the box.

In the unlikely event you specify both US and UK spellings of colour, the US spelling will take precedence.

Sometimes it can be useful to hide the outliers, for example when overlaying the raw data points on top of the boxplot. Hiding the outliers can be achieved by setting `outlier.shape = NA`. Importantly, this does not remove the outliers, it only hides them, so the range calculated for the y-axis will be the same with outliers shown and outliers hidden.

notch
If FALSE (default) make a standard box plot. If TRUE, make a notched box plot. Notches are used to compare groups; if the notches of two boxes do not overlap, this suggests that the medians are significantly different.

notchwidth
For a notched box plot, width of the notch relative to the body (defaults to `notchwidth = 0.5`).

varwidth
If FALSE (default) make a standard box plot. If TRUE, boxes are drawn with widths proportional to the square-roots of the number of observations in the groups (possibly weighted, using the `weight` aesthetic).

errorbar.draw
Draw horizontal whiskers at the top and bottom (the IQR). Defaults to ‘TRUE’.
geom_half_dotplot

- **errorbar.length**: Length of the horizontal whiskers (errorbar). Defaults to half the width of the half-boxplot.
- **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. borders().

**Examples**

```r
ggplot(iris, aes(x = Species, y = Petal.Width, fill = Species)) + geom_half_boxplot()
ggplot(iris, aes(x = Species, y = Petal.Width, fill = Species)) + geom_half_boxplot(side = "r")
ggplot(iris, aes(x = Species, y = Petal.Width, fill = Species)) + geom_half_boxplot(center = TRUE)
```

**Description**

In a dot plot, the width of a dot corresponds to the bin width (or maximum width, depending on the binning algorithm), and dots are stacked, with each dot representing one observation.

**Usage**

```r
geom_half_dotplot(
  mapping = NULL,
  data = NULL,
  position = "dodge",
  ..., 
  binwidth = NULL, 
  binaxis = "y",
  method = "dotdensity",
  binpositions = "bygroup",
  stackdir = "up",
  stackratio = 1,
  dotsize = 1,
  stackgroups = FALSE,
  origin = NULL,
)```
geom_half_dotplot

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. A function can be created from a formula (e.g. `~ head(.x, 10)`).

- **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 `colour = "red"` or `size = 3`. They may also be parameters to the paired geom/stat.

- **binwidth**
  When `method` is "dotdensity", this specifies maximum bin width. When `method` is "histodot", this specifies bin width. Defaults to 1/30 of the range of the data.

- **binaxis**
  The axis to bin along, "x" (default) or "y".

- **method**
  "dotdensity" (default) for dot-density binning, or "histodot" for fixed bin widths (like `stat_bin()`)

- **binpositions**
  When `method` is "dotdensity", "bygroup" (default) determines positions of the bins for each group separately. "all" determines positions of the bins with all the data taken together; this is used for aligning dot stacks across multiple groups.

- **stackdir**
  Which direction to stack the dots. "up" (default) places the half-dotplot on the right side. "down" on the left side.

- **stackratio**
  how close to stack the dots. Default is 1, where dots just touch. Use smaller values for closer, overlapping dots.

- **dotsize**
  The diameter of the dots relative to `binwidth`, default 1.

- **stackgroups**
  should dots be stacked across groups? This has the effect that `position = "stack"` should have, but can’t (because this geom has some odd properties).

- **origin**
  When `method` is "histodot", origin of first bin.

- **right**
  When `method` is "histodot", should intervals be closed on the right `(a, b)`, or not `[a, b)`
width: When binaxis is "y", the spacing of the dot stacks for dodging.

drop: If TRUE, remove all bins with zero counts.

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. borders().

Details

There are two basic approaches: dot-density and histodot. With dot-density binning, the bin positions are determined by the data and ‘binwidth’, which is the maximum width of each bin. See Wilkinson (1999) for details on the dot-density binning algorithm. With histodot binning, the bins have fixed positions and fixed widths, much like a histogram.

When binning along the x axis and stacking along the y axis, the numbers on y axis are not meaningful, due to technical limitations of ggplot2. You can hide the y axis, as in one of the examples, or manually scale it to match the number of dots.

Computed variables

- x: center of each bin, if binaxis is "x"
- y: center of each bin, if binaxis is "x"
- binwidth: max width of each bin if method is "dotdensity"; width of each bin if method is "histodot"
- count: number of points in bin
- ncount: count, scaled to maximum of 1
- density: density of points in bin, scaled to integrate to 1, if method is "histodot"
- ndensity: density, scaled to maximum of 1, if method is "histodot"

References


Examples

```r
ggplot(iris, aes(x = Species, y = Petal.Width, fill = Species)) +
  geom_half_dotplot(stackratio = .5, method = "histodot")
```
Description

Points with jitter for half geoms.

Usage

```r
gem half_point(
  mapping = NULL,
  data = NULL,
  stat = "HalfPoint",
  position = "dodge2",
  ..., 
  side = "r",
  transformation = position_jitter(),
  range_scale = 0.75,
  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. A function can be created from a `formula` (e.g. `~ head(.x, 10)`).
- `stat`: The statistical transformation to use on the data for this layer, as a string.
- `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 `colour = "red"` or `size = 3`. They may also be parameters to the paired geom/stat.
- `side`: The side on which to draw the half violin plot. "l" for left, "r" for right, defaults to "l".
transformation  An evaluated ‘position_*()’ function yielding a ‘Position’ object with specified parameters to calculate the transformation of the points. Defaults to ‘ggplot2::position_jitter()’.

range_scale  If no ‘width’ argument is specified in ‘transformation’, ‘range_scale’ is used to determine the width of the jitter. Defaults to ‘0.75’, which is half of the allotted space for the jitter-points, whereas ‘1’ would use all of the allotted space.

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. borders().

Examples

```r
ggplot(iris, aes(x = Species, y = Petal.Width, fill = Species)) + geom_half_point()
```

```r
ggplot(iris, aes(x = Species, y = Petal.Width, fill = Species)) + geom_half_point(side = "l")
```

description
Points with jitter for half geoms. Unlike ‘geom_half_point’, ‘geom_half_point_panel’ does not dodge different grouping aesthetics. This allows multiple groups in a single cloud of points (see examples).

Usage

```r
geom_half_point_panel(
  mapping = NULL,
  data = NULL,
  stat = "identity",
  position = "identity",
  ...,
  side = "r",
  transformation = position_jitter(),
  range_scale = 0.75,
)```
Arguments

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

data The data to be displayed in this layer. There are three options:

- If \texttt{NULL}, the default, the data is inherited from the plot data as specified in the call to \texttt{ggplot()}. A \texttt{data.frame}, or other object, will override the plot data. All objects will be fortified to produce a \texttt{data.frame}. See \texttt{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 \texttt{data.frame}, and will be used as the layer data. A function can be created from a formula (e.g. \texttt{~ head(.x, 10)}).

stat The statistical transformation to use on the data for this layer, as a string.

position Position adjustment, either as a string, or the result of a call to a position adjustment function.

... Other arguments passed on to \texttt{layer()}. These are often aesthetics, used to set an aesthetic to a fixed value, like \texttt{colour = “red”} or \texttt{size = 3}. They may also be parameters to the paired geom/stat.

side The side on which to draw the half violin plot. "l" for left, "r" for right, defaults to "l".

transformation An evaluated ‘position_*()’ function yielding a ‘Position’ object with specified parameters to calculate the transformation of the points. Defaults to ‘ggplot2::position_jitter()’.

range_scale If no ‘width’ argument is specified in ‘transformation’, ‘range_scale’ is used to determine the width of the jitter. Defaults to ‘0.75’, which is half of the allotted space for the jitter-points, whereas ‘1’ would use all of the allotted space.

na.rm If \texttt{FALSE}, the default, missing values are removed with a warning. If \texttt{TRUE}, missing values are silently removed.

show.legend logical. Should this layer be included in the legends? \texttt{NA}, the default, includes if any aesthetics are mapped. \texttt{FALSE} never includes, and \texttt{TRUE} always includes. It can also be a named logical vector to finely select the aesthetics to display.

inherit.aes If \texttt{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. \texttt{borders()}. Examples

\begin{verbatim}
    ggplot(iris, aes(y = Sepal.Width)) +
\end{verbatim}
geom_half_violin() +
geom_half_point_panel(aes(x = 0.5, color = Species), range_scale = .5) +
theme_classic()

---

**geom_half_violin**  
*Half Violin plot*

### Description
A violin plot is a compact display of a continuous distribution. It is a blend of [geom_boxplot()] and [geom_density()]: a violin plot is a mirrored density plot displayed in the same way as a boxplot.

### Usage
```r
geom_half_violin(
  mapping = NULL,
  data = NULL,
  stat = "half_ydensity",
  position = "dodge",
  ...,
  side = "l",
  nudge = 0,
  draw_quantiles = NULL,
  trim = TRUE,
  scale = "area",
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE
)
```

```r
stat_half_ydensity(
  mapping = NULL,
  data = NULL,
  geom = "half_violin",
  position = "dodge",
  ...,
  bw = "nrd0",
  adjust = 1,
  kernel = "gaussian",
  trim = TRUE,
  scale = "area",
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE
)
```
Arguments

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

data The data to be displayed in this layer. There are three options:
If \texttt{NULL}, the default, the data is inherited from the plot data as specified in the call to \texttt{ggplot()}.
A \texttt{data.frame}, or other object, will override the plot data. All objects will be fortified to produce a data frame. See \texttt{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 \texttt{data.frame}, and will be used as the layer data. A function can be created from a formula (e.g. \texttt{~ head(.x, 10)}).

position Position adjustment, either as a string, or the result of a call to a position adjustment function.

... Other arguments passed on to \texttt{layer()}. These are often aesthetics, used to set an aesthetic to a fixed value, like \texttt{colour = "red"} or \texttt{size = 3}. They may also be parameters to the paired geom/stat.

side The side on which to draw the half violin plot. "l" for left, "r" for right, defaults to "l".

nudge Add space between the violinplot and the middle of the space allotted to a given factor on the x-axis.

draw_quantiles If not (\texttt{NULL}) (default), draw horizontal lines at the given quantiles of the density estimate.

trim If \texttt{TRUE} (default), trim the tails of the violins to the range of the data. If \texttt{FALSE}, don’t trim the tails.

scale If "area" (default), all violins have the same area (before trimming the tails). If "count", areas are scaled proportionally to the number of observations. If "width", all violins have the same maximum width.

na.rm If \texttt{FALSE}, the default, missing values are removed with a warning. If \texttt{TRUE}, missing values are silently removed.

show.legend logical. Should this layer be included in the legends? \texttt{NA}, the default, includes if any aesthetics are mapped. \texttt{FALSE} never includes, and \texttt{TRUE} always includes. It can also be a named logical vector to finely select the aesthetics to display.

inherit.aes If \texttt{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. \texttt{borders()}.

gem, stat Use to override the default connection between \texttt{geom_violin()} and \texttt{stat_ydensity()}.

bw The smoothing bandwidth to be used. If numeric, the standard deviation of the smoothing kernel. If character, a rule to choose the bandwidth, as listed in \texttt{stats::bw.nrd()}.

adjust A multiplicative bandwidth adjustment. This makes it possible to adjust the bandwidth while still using the a bandwidth estimator. For example, \texttt{adjust = 1/2} means use half of the default bandwidth.

kernel Kernel. See list of available kernels in \texttt{density()}. 


Details

The half-violin plot accepts an optional ‘split’ aesthetic to compare data separated by a binary variable side-by-side.

References


See Also

[geom_half_violin()] for examples.

Examples

```r
ggplot(iris, aes(x = Species, y = Petal.Width, fill = Species)) + geom_half_violin()

ggplot(iris, aes(x = Species, y = Petal.Width, fill = Species)) + geom_half_violin(side = "r")

ggplot() + geom_half_violin(
  data = ToothGrowth,
  aes(x = as.factor(dose), y = len, split = supp, fill = supp),
  position = "identity"
) + theme_minimal()

ggplot(ToothGrowth, aes(x = supp, y = len, color = supp)) + geom_half_violin(side = c("l", "r"))

ggplot(data = ToothGrowth, aes(x = 1, y = len)) + geom_half_point(aes(y = len), side = "l") + geom_half_violin(aes(y = len), side = "r") + coord_flip()
```
Index

* datasets
  GeomHalfDotplot, 2

aes(), 3, 5, 7, 9, 11
aes_(), 3, 5, 7, 9, 11

borders(), 4, 6, 8, 9, 11
density(), 11

fortify(), 3, 5, 7, 9, 11

geom_half_boxplot, 2
geom_half_dotplot, 4
geom_half_point, 7
geom_half_point_panel, 8
geom_half_violin, 10
GeomHalfBoxplot (GeomHalfDotplot), 2
GeomHalfDotplot, 2
GeomHalfPoint (GeomHalfDotplot), 2
GeomHalfPointPanel (GeomHalfDotplot), 2
GeomHalfViolin (GeomHalfDotplot), 2
gghalves-extensions (GeomHalfDotplot), 2
ggplot(), 3, 5, 7, 9, 11

layer(), 3, 5, 7, 9, 11

stat_half_ydensity (geom_half_violin), 10
StatHalfBindot (GeomHalfDotplot), 2
StatHalfPoint (GeomHalfDotplot), 2
StatHalfYdensity (GeomHalfDotplot), 2
stats::bw.nrd(), 11