Package 'ggdensity'

May 8, 2026

Title Interpretable Bivariate Density Visualization with 'ggplot2'

Version 1.0.1

Description The 'ggplot2' package provides simple functions for visualizing contours of 2-d kernel density estimates. 'ggdensity' implements several additional density estimators as well as more interpretable visualizations based on highest density regions instead of the traditional height of the estimated density surface.

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.3.3

Depends ggplot2

Imports isoband, vctrs, tibble, MASS, stats, scales

URL https://jamesotto852.github.io/ggdensity/, https://github.com/jamesotto852/ggdensity/

BugReports https://github.com/jamesotto852/ggdensity/issues/

Suggests vdiffr, testthat (>= 3.0.0), knitr, rmarkdown

Config/testthat/edition 3

VignetteBuilder knitr

NeedsCompilation no

Author James Otto [aut, cre, cph] (ORCID: <https://orcid.org/0000-0002-0665-2515>), David Kahle [aut] (ORCID: <https://orcid.org/0000-0002-9999-1558>)

Maintainer James Otto <jamesotto852@gmail.com>

Repository CRAN

Date/Publication 2026-02-26 06:10:12 UTC

Contents

`geom_hdr` *Highest density regions of a 2D density estimate*

Description

Perform 2D density estimation, compute and plot the resulting highest density regions. `geom_hdr()` draws filled regions and `geom_hdr_lines()` draws lines outlining the regions. Note, the plotted objects have probabilities mapped to the `alpha` aesthetic by default.

Usage

```
stat_hdr(
  mapping = NULL,
  data = NULL,
  geom = "hdr",
  position = "identity",
  ...,
  method = "kde",
  probs = c(0.99, 0.95, 0.8, 0.5),
  n = 100,
  xlim = NULL,
  ylim = NULL,
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE
)

geom_hdr(
  mapping = NULL,
  data = NULL,
```

```
  stat = "hdr",
  position = "identity",
  ...,
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE
)
```

Arguments

mapping	Set of aesthetic mappings created by 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)).
geom	The geometric object to use to display the data for this layer. When using a stat_*() function to construct a layer, the geom argument can be used to override the default coupling between stats and geoms. The geom argument accepts the following: • A Geom ggproto subclass, for example GeomPoint. • A string naming the geom. To give the geom as a string, strip the function name of the geom_ prefix. For example, to use geom_point(), give the geom as "point". • For more information and other ways to specify the geom, see the layer geom documentation.
position	A position adjustment to use on the data for this layer. This can be used in various ways, including to prevent overplotting and improving the display. The position argument accepts the following: • The result of calling a position function, such as position_jitter(). This method allows for passing extra arguments to the position. • A string naming the position adjustment. To give the position as a string, strip the function name of the position_ prefix. For example, to use position_jitter(), give the position as "jitter". • For more information and other ways to specify the position, see the layer position documentation.
...	Other arguments passed on to layer()'s params argument. These arguments broadly fall into one of 4 categories below. Notably, further arguments to the position argument, or aesthetics that are required can *not* be passed through .... Unknown arguments that are not part of the 4 categories below are ignored.

- Static aesthetics that are not mapped to a scale, but are at a fixed value and apply to the layer as a whole. For example, `colour = "red"` or `linewidth = 3`. The geom's documentation has an **Aesthetics** section that lists the available options. The 'required' aesthetics cannot be passed on to the `params`. Please note that while passing unmapped aesthetics as vectors is technically possible, the order and required length is not guaranteed to be parallel to the input data.
- When constructing a layer using a `stat_*()` function, the `...` argument can be used to pass on parameters to the `geom` part of the layer. An example of this is `stat_density(geom = "area", outline.type = "both")`. The geom's documentation lists which parameters it can accept.
- Inversely, when constructing a layer using a `geom_*()` function, the `...` argument can be used to pass on parameters to the `stat` part of the layer. An example of this is `geom_area(stat = "density", adjust = 0.5)`. The stat's documentation lists which parameters it can accept.
- The `key_glyph` argument of `layer()` may also be passed on through `....` This can be one of the functions described as key glyphs, to change the display of the layer in the legend.

`method` Density estimator to use, accepts character vector: `"kde"`,`"histogram"`, `"freqpoly"`, or `"mvnorm"`. Alternatively accepts functions which return closures corresponding to density estimates, see `?get_hdr` or `vignette("method", "ggdensity")`.

`probs` Probabilities to compute highest density regions for.

`n` Resolution of grid defined by `xlim` and `ylim`. Ignored if `method = "histogram"` or `method = "freqpoly"`.

`xlim, ylim` Range to compute and draw regions. If `NULL`, defaults to range of data.

`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. To include legend keys for all levels, even when no data exists, use `TRUE`. If `NA`, all levels are shown in legend, but unobserved levels are omitted.

`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. `annotation_borders()`.

`stat` The statistical transformation to use on the data for this layer. When using a `geom_*()` function to construct a layer, the `stat` argument can be used to override the default coupling between geoms and stats. The `stat` argument accepts the following:

- A `Stat` ggproto subclass, for example `StatCount`.
- A string naming the stat. To give the stat as a string, strip the function name of the `stat_` prefix. For example, to use `stat_count()`, give the stat as `"count"`.
- For more information and other ways to specify the stat, see the layer stat documentation.

Aesthetics

geom_hdr() and geom_hdr_lines() understand the following aesthetics (required aesthetics are in bold):

- **x**
- **y**
- alpha
- color
- fill (only geom_hdr)
- group
- linetype
- linewidth
- subgroup

Computed variables

probs The probability associated with the highest density region, specified by probs argument.

References

Scott, David W. Multivariate Density Estimation (2e), Wiley.

Examples

```
# Basic simulated data with bivariate normal data and various methods
df <- data.frame(x = rnorm(1000), y = rnorm(1000))
p <- ggplot(df, aes(x, y)) + coord_equal()

p + geom_hdr()
p + geom_hdr(method = "mvnorm")
p + geom_hdr(method = "freqpoly")
# p + geom_hdr(method = "histogram")

# Adding point layers on top to visually assess region estimates
pts <- geom_point(size = .2, color = "red")

p + geom_hdr() + pts
p + geom_hdr(method = "mvnorm") + pts
p + geom_hdr(method = "freqpoly") + pts
# p + geom_hdr(method = "histogram") + pts

# Highest density region boundary lines
p + geom_hdr_lines()
p + geom_hdr_lines(method = "mvnorm")
p + geom_hdr_lines(method = "freqpoly")
# p + geom_hdr_lines(method = "histogram")

## Not run:
```

```
# 2+ groups - mapping other aesthetics in the geom
rdata <- function(n, n_groups = 3, radius = 3) {
  list_of_dfs <- lapply(0:(n_groups-1), function(k) {
    mu <- c(cos(2*k*pi/n_groups), sin(2*k*pi/n_groups))
    m <- MASS::mvrnorm(n, radius*mu, diag(2))
    structure(data.frame(m, as.character(k)), names = c("x", "y", "c"))
  })
  do.call("rbind", list_of_dfs)
}

dfc <- rdata(1000, n_groups = 5)
pf <- ggplot(dfc, aes(x, y, fill = c)) + coord_equal()

pf + geom_hdr()
pf + geom_hdr(method = "mvnorm")
pf + geom_hdr(method = "mvnorm", probs = .90, alpha = .5)
pf + geom_hdr(method = "histogram")
pf + geom_hdr(method = "freqpoly")

pc <- ggplot(dfc, aes(x, y, color = c)) +
 coord_equal() +
 theme_minimal() +
 theme(panel.grid.minor = element_blank())

pc + geom_hdr_lines()
pc + geom_hdr_lines(method = "mvnorm")


# Data with boundaries
ggplot(df, aes(x^2)) + geom_histogram(bins = 30)
ggplot(df, aes(x^2)) + geom_histogram(bins = 30, boundary = 0)
ggplot(df, aes(x^2, y^2)) + geom_hdr(method = "histogram")


## End(Not run)
```

`geom_hdr_fun` *Highest density regions of a bivariate pdf*

Description

Compute and plot the highest density regions (HDRs) of a bivariate pdf. `geom_hdr_fun()` draws filled regions, and `geom_hdr_lines_fun()` draws lines outlining the regions. Note, the plotted objects have probabilities mapped to the `alpha` aesthetic by default.

Usage

```
stat_hdr_fun(
```

```
  mapping = NULL,
  data = NULL,
  geom = "hdr_fun",
  position = "identity",
  ...,
  fun,
  args = list(),
  probs = c(0.99, 0.95, 0.8, 0.5),
  xlim = NULL,
  ylim = NULL,
  n = 100,
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE
)

geom_hdr_fun(
  mapping = NULL,
  data = NULL,
  stat = "hdr_fun",
  position = "identity",
  ...,
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE
)
```

Arguments

mapping	Set of aesthetic mappings created by 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)).
geom	The geometric object to use to display the data for this layer. When using a stat_*() function to construct a layer, the geom argument can be used to override the default coupling between stats and geoms. The geom argument accepts the following: • A Geom ggproto subclass, for example GeomPoint.

- A string naming the geom. To give the geom as a string, strip the function name of the `geom_` prefix. For example, to use `geom_point()`, give the geom as `"point"`.
- For more information and other ways to specify the geom, see the layer geom documentation.

`position` A position adjustment to use on the data for this layer. This can be used in various ways, including to prevent overplotting and improving the display. The `position` argument accepts the following:

- The result of calling a position function, such as `position_jitter()`. This method allows for passing extra arguments to the position.
- A string naming the position adjustment. To give the position as a string, strip the function name of the `position_` prefix. For example, to use `position_jitter()`, give the position as `"jitter"`.
- For more information and other ways to specify the position, see the layer position documentation.

`...` Other arguments passed on to `layer()`'s `params` argument. These arguments broadly fall into one of 4 categories below. Notably, further arguments to the `position` argument, or aesthetics that are required can *not* be passed through `....` Unknown arguments that are not part of the 4 categories below are ignored.

- Static aesthetics that are not mapped to a scale, but are at a fixed value and apply to the layer as a whole. For example, `colour = "red"` or `linewidth = 3`. The geom's documentation has an **Aesthetics** section that lists the available options. The 'required' aesthetics cannot be passed on to the `params`. Please note that while passing unmapped aesthetics as vectors is technically possible, the order and required length is not guaranteed to be parallel to the input data.
- When constructing a layer using a `stat_*()` function, the `...` argument can be used to pass on parameters to the `geom` part of the layer. An example of this is `stat_density(geom = "area", outline.type = "both")`. The geom's documentation lists which parameters it can accept.
- Inversely, when constructing a layer using a `geom_*()` function, the `...` argument can be used to pass on parameters to the `stat` part of the layer. An example of this is `geom_area(stat = "density", adjust = 0.5)`. The stat's documentation lists which parameters it can accept.
- The `key_glyph` argument of `layer()` may also be passed on through `....` This can be one of the functions described as key glyphs, to change the display of the layer in the legend.

`fun` A function, the joint probability density function, must be vectorized in its first two arguments; see examples.

`args` Named list of additional arguments passed on to `fun`.

`probs` Probabilities to compute highest density regions for.

`xlim, ylim` Range to compute and draw regions. If `NULL`, defaults to range of data if present.

`n` Resolution of grid `fun` is evaluated on.

`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. To include legend keys for all levels, even when no data exists, use TRUE. If NA, all levels are shown in legend, but unobserved levels are omitted.

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. annotation_borders().

stat — The statistical transformation to use on the data for this layer. When using a geom_*() function to construct a layer, the stat argument can be used to override the default coupling between geoms and stats. The stat argument accepts the following:

- A Stat ggproto subclass, for example StatCount.
- A string naming the stat. To give the stat as a string, strip the function name of the stat_ prefix. For example, to use stat_count(), give the stat as "count".
- For more information and other ways to specify the stat, see the layer stat documentation.

Aesthetics

geom_hdr_fun() and geom_hdr_lines_fun() understand the following aesthetics (required aesthetics are in bold):

- x
- y
- alpha
- color
- fill (only geom_hdr_fun)
- group
- linetype
- linewidth
- subgroup

Computed variables

probs The probability associated with the highest density region, specified by probs.

Examples

```
# HDRs of the bivariate exponential
f <- function(x, y) dexp(x) * dexp(y)
ggplot() + geom_hdr_fun(fun = f, xlim = c(0, 10), ylim = c(0, 10))


# HDRs of a custom parametric model
```

```
# generate example data
n <- 1000
th_true <- c(3, 8)

rdata <- function(n, th) {
  gen_single_obs <- function(th) {
    rchisq(2, df = th) # can be anything
  }
  df <- replicate(n, gen_single_obs(th))
  setNames(as.data.frame(t(df)), c("x", "y"))
}
data <- rdata(n, th_true)

# estimate unknown parameters via maximum likelihood
likelihood <- function(th) {
  th <- abs(th) # hack to enforce parameter space boundary
  log_f <- function(v) {
    x <- v[1]; y <- v[2]
    dchisq(x, df = th[1], log = TRUE) + dchisq(y, df = th[2], log = TRUE)
  }
  sum(apply(data, 1, log_f))
}
(th_hat <- optim(c(1, 1), likelihood, control = list(fnscale = -1))$par)

# plot f for the give model
f <- function(x, y, th) dchisq(x, df = th[1]) * dchisq(y, df = th[2])

ggplot(data, aes(x, y)) +
  geom_hdr_fun(fun = f, args = list(th = th_hat)) +
  geom_point(size = .25, color = "red") +
  xlim(0, 30) + ylim(c(0, 30))

ggplot(data, aes(x, y)) +
  geom_hdr_lines_fun(fun = f, args = list(th = th_hat)) +
  geom_point(size = .25, color = "red") +
  xlim(0, 30) + ylim(c(0, 30))
```

`geom_hdr_points` *Scatterplot colored by highest density regions of a 2D density estimate*

Description

Perform 2D density estimation, compute the resulting highest density regions (HDRs), and plot the provided data as a scatterplot with points colored according to their corresponding HDR.

Usage

```
stat_hdr_points(
  mapping = NULL,
  data = NULL,
  geom = "point",
  position = "identity",
  ...,
  method = "kde",
  probs = c(0.99, 0.95, 0.8, 0.5),
  n = 100,
  xlim = NULL,
  ylim = NULL,
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE
)

geom_hdr_points(
  mapping = NULL,
  data = NULL,
  stat = "hdr_points",
  position = "identity",
  ...,
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE
)
```

Arguments

mapping	Set of aesthetic mappings created by 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)).
geom	The geometric object to use to display the data for this layer. When using a stat_*() function to construct a layer, the geom argument can be used to override the default coupling between stats and geoms. The geom argument accepts the following: • A Geom ggproto subclass, for example GeomPoint.

- A string naming the geom. To give the geom as a string, strip the function name of the geom_ prefix. For example, to use geom_point(), give the geom as "point".
- For more information and other ways to specify the geom, see the layer geom documentation.

position — A position adjustment to use on the data for this layer. This can be used in various ways, including to prevent overplotting and improving the display. The position argument accepts the following:

- The result of calling a position function, such as position_jitter(). This method allows for passing extra arguments to the position.
- A string naming the position adjustment. To give the position as a string, strip the function name of the position_ prefix. For example, to use position_jitter(), give the position as "jitter".
- For more information and other ways to specify the position, see the layer position documentation.

... — Other arguments passed on to layer()'s params argument. These arguments broadly fall into one of 4 categories below. Notably, further arguments to the position argument, or aesthetics that are required can *not* be passed through .... Unknown arguments that are not part of the 4 categories below are ignored.

- Static aesthetics that are not mapped to a scale, but are at a fixed value and apply to the layer as a whole. For example, colour = "red" or linewidth = 3. The geom's documentation has an **Aesthetics** section that lists the available options. The 'required' aesthetics cannot be passed on to the params. Please note that while passing unmapped aesthetics as vectors is technically possible, the order and required length is not guaranteed to be parallel to the input data.
- When constructing a layer using a stat_*() function, the ... argument can be used to pass on parameters to the geom part of the layer. An example of this is stat_density(geom = "area", outline.type = "both"). The geom's documentation lists which parameters it can accept.
- Inversely, when constructing a layer using a geom_*() function, the ... argument can be used to pass on parameters to the stat part of the layer. An example of this is geom_area(stat = "density", adjust = 0.5). The stat's documentation lists which parameters it can accept.
- The key_glyph argument of layer() may also be passed on through .... This can be one of the functions described as key glyphs, to change the display of the layer in the legend.

method — Density estimator to use, accepts character vector: "kde","histogram", "freqpoly", or "mvnorm". Alternatively accepts functions which return closures corresponding to density estimates, see ?get_hdr or vignette("method", "ggdensity").

probs — Probabilities to compute highest density regions for.

n — Number of grid points in each direction.

xlim, ylim — Range to compute and draw regions. If NULL, defaults to range of data.

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. To include legend keys for all levels, even when no data exists, use TRUE. If NA, all levels are shown in legend, but unobserved levels are omitted.

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. annotation_borders().

stat — The statistical transformation to use on the data for this layer. When using a geom_*() function to construct a layer, the stat argument can be used to override the default coupling between geoms and stats. The stat argument accepts the following:

- A Stat ggproto subclass, for example StatCount.
- A string naming the stat. To give the stat as a string, strip the function name of the stat_ prefix. For example, to use stat_count(), give the stat as "count".
- For more information and other ways to specify the stat, see the layer stat documentation.

Aesthetics

geom_hdr_points understands the following aesthetics (required aesthetics are in bold):

- **x**
- **y**
- alpha
- color
- fill
- group
- linetype
- size
- subgroup

Computed variables

probs The probability associated with the highest density region, specified by probs.

Examples

```
set.seed(1)
df <- data.frame(x = rnorm(500), y = rnorm(500))
p <- ggplot(df, aes(x, y)) +
 coord_equal()

p + geom_hdr_points()
```

```
# Setting aes(fill = after_stat(probs)), color = "black", and
# shape = 21 helps alleviate overplotting:
p + geom_hdr_points(aes(fill = after_stat(probs)), color = "black", shape = 21, size = 2)

# Also works well with geom_hdr_lines()
p +
 geom_hdr_lines(
   aes(color = after_stat(probs)), alpha = 1,
   xlim = c(-5, 5), ylim = c(-5, 5)
 ) +
 geom_hdr_points(
   aes(fill = after_stat(probs)), color = "black", shape = 21, size = 2,
   xlim = c(-5, 5), ylim = c(-5, 5)
 )
```

`geom_hdr_points_fun` *Scatterplot colored by highest density regions of a bivariate pdf*

Description

Compute the highest density regions (HDRs) of a bivariate pdf and plot the provided data as a scatterplot with points colored according to their corresponding HDR.

Usage

```
stat_hdr_points_fun(
  mapping = NULL,
  data = NULL,
  geom = "point",
  position = "identity",
  ...,
  fun,
  args = list(),
  probs = c(0.99, 0.95, 0.8, 0.5),
  xlim = NULL,
  ylim = NULL,
  n = 100,
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE
)

geom_hdr_points_fun(
  mapping = NULL,
  data = NULL,
  stat = "hdr_points_fun",
  position = "identity",
```

```
  ...,
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE
)
```

Arguments

mapping	Set of aesthetic mappings created by 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)).
geom	The geometric object to use to display the data for this layer. When using a stat_*() function to construct a layer, the geom argument can be used to override the default coupling between stats and geoms. The geom argument accepts the following: • A Geom ggproto subclass, for example GeomPoint. • A string naming the geom. To give the geom as a string, strip the function name of the geom_ prefix. For example, to use geom_point(), give the geom as "point". • For more information and other ways to specify the geom, see the layer geom documentation.
position	A position adjustment to use on the data for this layer. This can be used in various ways, including to prevent overplotting and improving the display. The position argument accepts the following: • The result of calling a position function, such as position_jitter(). This method allows for passing extra arguments to the position. • A string naming the position adjustment. To give the position as a string, strip the function name of the position_ prefix. For example, to use position_jitter(), give the position as "jitter". • For more information and other ways to specify the position, see the layer position documentation.
...	Other arguments passed on to layer()'s params argument. These arguments broadly fall into one of 4 categories below. Notably, further arguments to the position argument, or aesthetics that are required can *not* be passed through .... Unknown arguments that are not part of the 4 categories below are ignored. • Static aesthetics that are not mapped to a scale, but are at a fixed value and apply to the layer as a whole. For example, colour = "red" or linewidth

= 3. The geom's documentation has an **Aesthetics** section that lists the available options. The 'required' aesthetics cannot be passed on to the params. Please note that while passing unmapped aesthetics as vectors is technically possible, the order and required length is not guaranteed to be parallel to the input data.

- When constructing a layer using a stat_*() function, the ... argument can be used to pass on parameters to the geom part of the layer. An example of this is stat_density(geom = "area", outline.type = "both"). The geom's documentation lists which parameters it can accept.
- Inversely, when constructing a layer using a geom_*() function, the ... argument can be used to pass on parameters to the stat part of the layer. An example of this is geom_area(stat = "density", adjust = 0.5). The stat's documentation lists which parameters it can accept.
- The key_glyph argument of layer() may also be passed on through .... This can be one of the functions described as key glyphs, to change the display of the layer in the legend.

fun — A function, the joint probability density function, must be vectorized in its first two arguments; see examples.

args — Named list of additional arguments passed on to fun.

probs — Probabilities to compute highest density regions for.

xlim, ylim — Range to compute and draw regions. If NULL, defaults to range of data if present.

n — Number of grid points in each direction.

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. To include legend keys for all levels, even when no data exists, use TRUE. If NA, all levels are shown in legend, but unobserved levels are omitted.

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. annotation_borders().

stat — The statistical transformation to use on the data for this layer. When using a geom_*() function to construct a layer, the stat argument can be used to override the default coupling between geoms and stats. The stat argument accepts the following:

- A Stat ggproto subclass, for example StatCount.
- A string naming the stat. To give the stat as a string, strip the function name of the stat_ prefix. For example, to use stat_count(), give the stat as "count".
- For more information and other ways to specify the stat, see the layer stat documentation.

Aesthetics

geom_hdr_points_fun understands the following aesthetics (required aesthetics are in bold):

- **x**
- **y**
- alpha
- color
- fill
- group
- linetype
- size
- subgroup

Computed variables

probs The probability associated with the highest density region, specified by probs.

Examples

```
# Can plot points colored according to known pdf:
set.seed(1)
df <- data.frame(x = rexp(1000), y = rexp(1000))
f <- function(x, y) dexp(x) * dexp(y)

ggplot(df, aes(x, y)) +
  geom_hdr_points_fun(fun = f, xlim = c(0, 10), ylim = c(0, 10))


# Also allows for hdrs of a custom parametric model

# generate example data
n <- 1000
th_true <- c(3, 8)

rdata <- function(n, th) {
  gen_single_obs <- function(th) {
    rchisq(2, df = th) # can be anything
  }
  df <- replicate(n, gen_single_obs(th))
  setNames(as.data.frame(t(df)), c("x", "y"))
}
data <- rdata(n, th_true)

# estimate unknown parameters via maximum likelihood
likelihood <- function(th) {
  th <- abs(th) # hack to enforce parameter space boundary
  log_f <- function(v) {
    x <- v[1]; y <- v[2]
    dchisq(x, df = th[1], log = TRUE) + dchisq(y, df = th[2], log = TRUE)
```

```
  }
  sum(apply(data, 1, log_f))
}
(th_hat <- optim(c(1, 1), likelihood, control = list(fnscale = -1))$par)

# plot f for the give model
f <- function(x, y, th) dchisq(x, df = th[1]) * dchisq(y, df = th[2])

ggplot(data, aes(x, y)) +
  geom_hdr_points_fun(fun = f, args = list(th = th_hat))

ggplot(data, aes(x, y)) +
  geom_hdr_points_fun(aes(fill = after_stat(probs)), shape = 21, color = "black",
    fun = f, args = list(th = th_hat), na.rm = TRUE) +
 geom_hdr_lines_fun(aes(color = after_stat(probs)), alpha = 1, fun = f, args = list(th = th_hat)) +
  lims(x = c(0, 15), y = c(0, 25))
```

---

geom_hdr_rug *Rug plots of marginal highest density region estimates*

Description

Perform 1D density estimation, compute and plot the resulting highest density regions in a way similar to ggplot2::geom_rug(). Note, the plotted objects have probabilities mapped to the alpha aesthetic by default.

Usage

```
stat_hdr_rug(
  mapping = NULL,
  data = NULL,
  geom = "hdr_rug",
  position = "identity",
  ...,
  method = "kde",
  method_y = "kde",
  probs = c(0.99, 0.95, 0.8, 0.5),
  xlim = NULL,
  ylim = NULL,
  n = 512,
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE
)

geom_hdr_rug(
  mapping = NULL,
```

```
  data = NULL,
  stat = "hdr_rug",
  position = "identity",
  ...,
  outside = FALSE,
  sides = "bl",
  length = unit(0.03, "npc"),
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE
)
```

Arguments

mapping	Set of aesthetic mappings created by `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)`).
geom	The geometric object to use to display the data for this layer. When using a `stat_*()` function to construct a layer, the `geom` argument can be used to override the default coupling between stats and geoms. The `geom` argument accepts the following: • A `Geom` ggproto subclass, for example `GeomPoint`. • A string naming the geom. To give the geom as a string, strip the function name of the `geom_` prefix. For example, to use `geom_point()`, give the geom as `"point"`. • For more information and other ways to specify the geom, see the layer geom documentation.
position	A position adjustment to use on the data for this layer. This can be used in various ways, including to prevent overplotting and improving the display. The `position` argument accepts the following: • The result of calling a position function, such as `position_jitter()`. This method allows for passing extra arguments to the position. • A string naming the position adjustment. To give the position as a string, strip the function name of the `position_` prefix. For example, to use `position_jitter()`, give the position as `"jitter"`. • For more information and other ways to specify the position, see the layer position documentation.

`...` Other arguments passed on to `layer()`'s `params` argument. These arguments broadly fall into one of 4 categories below. Notably, further arguments to the `position` argument, or aesthetics that are required can *not* be passed through `...`. Unknown arguments that are not part of the 4 categories below are ignored.

- Static aesthetics that are not mapped to a scale, but are at a fixed value and apply to the layer as a whole. For example, `colour = "red"` or `linewidth = 3`. The geom's documentation has an **Aesthetics** section that lists the available options. The 'required' aesthetics cannot be passed on to the `params`. Please note that while passing unmapped aesthetics as vectors is technically possible, the order and required length is not guaranteed to be parallel to the input data.
- When constructing a layer using a `stat_*()` function, the `...` argument can be used to pass on parameters to the `geom` part of the layer. An example of this is `stat_density(geom = "area", outline.type = "both")`. The geom's documentation lists which parameters it can accept.
- Inversely, when constructing a layer using a `geom_*()` function, the `...` argument can be used to pass on parameters to the `stat` part of the layer. An example of this is `geom_area(stat = "density", adjust = 0.5)`. The stat's documentation lists which parameters it can accept.
- The `key_glyph` argument of `layer()` may also be passed on through `...`. This can be one of the functions described as key glyphs, to change the display of the layer in the legend.

`method, method_y` Density estimator(s) to use. By default `method` is used for both x- and y-axis. If specified, `method_y` will be used for y-axis. Accepts character vector: `"kde"`,`"histogram"`, `"freqpoly"`, or `"norm"`. Alternatively accepts functions which return closures corresponding to density estimates, see `?get_hdr_1d` or `vignette("method", "ggdensity")`.

`probs` Probabilities to compute highest density regions for.

`xlim, ylim` Range to compute and draw regions. If `NULL`, defaults to range of data.

`n` Resolution of grid defined by `xlim` and `ylim`. Ignored if `method = "histogram"` or `method = "freqpoly"`.

`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. To include legend keys for all levels, even when no data exists, use `TRUE`. If `NA`, all levels are shown in legend, but unobserved levels are omitted.

`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. `annotation_borders()`.

`stat` The statistical transformation to use on the data for this layer. When using a `geom_*()` function to construct a layer, the `stat` argument can be used to override the default coupling between geoms and stats. The `stat` argument accepts the following:

- A `Stat` ggproto subclass, for example `StatCount`.
- A string naming the stat. To give the stat as a string, strip the function name of the `stat_` prefix. For example, to use `stat_count()`, give the stat as `"count"`.
- For more information and other ways to specify the stat, see the layer stat documentation.

`outside` logical that controls whether to move the rug tassels outside of the plot area. Default is off (FALSE). You will also need to use `coord_cartesian(clip = "off")`. When set to TRUE, also consider changing the sides argument to "tr". See examples.

`sides` A string that controls which sides of the plot the rugs appear on. It can be set to a string containing any of `"trbl"`, for top, right, bottom, and left.

`length` A `grid::unit()` object that sets the length of the rug lines. Use scale expansion to avoid overplotting of data.

Aesthetics

geom_hdr_rug understands the following aesthetics (required aesthetics are in bold):

- x
- y
- alpha
- fill
- group
- subgroup

Computed variables

probs The probability of the highest density region, specified by `probs`, corresponding to each point.

Examples

```
set.seed(1)
df <- data.frame(x = rnorm(100), y = rnorm(100))

# Plot marginal HDRs for bivariate data
ggplot(df, aes(x, y)) +
  geom_point() +
  geom_hdr_rug() +
  coord_fixed()

ggplot(df, aes(x, y)) +
  geom_hdr() +
  geom_hdr_rug() +
  coord_fixed()

# Plot HDR for univariate data
```

```
ggplot(df, aes(x)) +
  geom_density() +
  geom_hdr_rug()

ggplot(df, aes(y = y)) +
  geom_density() +
  geom_hdr_rug()

# Specify location of marginal HDRs as in ggplot2::geom_rug()
ggplot(df, aes(x, y)) +
  geom_hdr() +
  geom_hdr_rug(sides = "tr", outside = TRUE) +
  coord_fixed(clip = "off")

# Can use same methods of density estimation as geom_hdr().
# For data with constrained support, we suggest setting method = "histogram":
ggplot(df, aes(x^2)) +
 geom_histogram(bins = 30, boundary = 0) +
 geom_hdr_rug(method = "histogram")

ggplot(df, aes(x^2, y^2)) +
 geom_hdr(method = "histogram") +
 geom_hdr_rug(method = "histogram") +
 coord_fixed()
```

---

geom_hdr_rug_fun *Rug plots of highest density region estimates of univariate pdfs*

Description

Compute and plot the highest density regions (HDRs) of specified univariate pdf(s). Note, the plotted objects have probabilities mapped to the `alpha` aesthetic by default.

Usage

```
stat_hdr_rug_fun(
  mapping = NULL,
  data = NULL,
  geom = "hdr_rug_fun",
  position = "identity",
  ...,
  fun_x = NULL,
  fun_y = NULL,
  args_x = list(),
  args_y = list(),
  probs = c(0.99, 0.95, 0.8, 0.5),
  xlim = NULL,
  ylim = NULL,
```

```
  n = 512,
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE
)

geom_hdr_rug_fun(
  mapping = NULL,
  data = NULL,
  stat = "hdr_rug_fun",
  position = "identity",
  ...,
  outside = FALSE,
  sides = "bl",
  length = unit(0.03, "npc"),
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE
)
```

Arguments

mapping	Set of aesthetic mappings created by 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)).
geom	The geometric object to use to display the data for this layer. When using a stat_*() function to construct a layer, the geom argument can be used to override the default coupling between stats and geoms. The geom argument accepts the following: • A Geom ggproto subclass, for example GeomPoint. • A string naming the geom. To give the geom as a string, strip the function name of the geom_ prefix. For example, to use geom_point(), give the geom as "point". • For more information and other ways to specify the geom, see the layer geom documentation.
position	A position adjustment to use on the data for this layer. This can be used in various ways, including to prevent overplotting and improving the display. The position argument accepts the following:

- The result of calling a position function, such as `position_jitter()`. This method allows for passing extra arguments to the position.
- A string naming the position adjustment. To give the position as a string, strip the function name of the `position_` prefix. For example, to use `position_jitter()`, give the position as `"jitter"`.
- For more information and other ways to specify the position, see the layer position documentation.

`...` Other arguments passed on to `layer()`'s `params` argument. These arguments broadly fall into one of 4 categories below. Notably, further arguments to the `position` argument, or aesthetics that are required can *not* be passed through `....` Unknown arguments that are not part of the 4 categories below are ignored.

- Static aesthetics that are not mapped to a scale, but are at a fixed value and apply to the layer as a whole. For example, `colour = "red"` or `linewidth = 3`. The geom's documentation has an **Aesthetics** section that lists the available options. The 'required' aesthetics cannot be passed on to the `params`. Please note that while passing unmapped aesthetics as vectors is technically possible, the order and required length is not guaranteed to be parallel to the input data.
- When constructing a layer using a `stat_*()` function, the `...` argument can be used to pass on parameters to the `geom` part of the layer. An example of this is `stat_density(geom = "area", outline.type = "both")`. The geom's documentation lists which parameters it can accept.
- Inversely, when constructing a layer using a `geom_*()` function, the `...` argument can be used to pass on parameters to the `stat` part of the layer. An example of this is `geom_area(stat = "density", adjust = 0.5)`. The stat's documentation lists which parameters it can accept.
- The `key_glyph` argument of `layer()` may also be passed on through `....` This can be one of the functions described as key glyphs, to change the display of the layer in the legend.

`fun_x, fun_y` Functions, the univariate probability density function for the x- and/or y-axis. First argument must be vectorized.

`args_x, args_y` Named list of additional arguments passed on to `fun_x` and/or `fun_y`.

`probs` Probabilities to compute highest density regions for.

`xlim, ylim` Range to compute and draw regions. If `NULL`, defaults to range of data.

`n` Resolution of grid defined by `xlim` and `ylim`. Ignored if `method = "histogram"` or `method = "freqpoly"`.

`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. To include legend keys for all levels, even when no data exists, use `TRUE`. If `NA`, all levels are shown in legend, but unobserved levels are omitted.

`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. `annotation_borders()`.

`stat` The statistical transformation to use on the data for this layer. When using a `geom_*()` function to construct a layer, the `stat` argument can be used to override the default coupling between geoms and stats. The `stat` argument accepts the following:

- A `Stat` ggproto subclass, for example `StatCount`.
- A string naming the stat. To give the stat as a string, strip the function name of the `stat_` prefix. For example, to use `stat_count()`, give the stat as `"count"`.
- For more information and other ways to specify the stat, see the layer stat documentation.

`outside` logical that controls whether to move the rug tassels outside of the plot area. Default is off (FALSE). You will also need to use `coord_cartesian(clip = "off")`. When set to TRUE, also consider changing the sides argument to "tr". See examples.

`sides` A string that controls which sides of the plot the rugs appear on. It can be set to a string containing any of `"trbl"`, for top, right, bottom, and left.

`length` A `grid::unit()` object that sets the length of the rug lines. Use scale expansion to avoid overplotting of data.

Aesthetics

`geom_hdr_rug_fun()` understands the following aesthetics (required aesthetics are in bold):

- x
- y
- alpha
- fill
- group
- subgroup

Computed variables

probs The probability of the highest density region, specified by `probs`, corresponding to each point.

Examples

```
# Plotting data with exponential marginals
df <- data.frame(x = rexp(1e3), y = rexp(1e3))

ggplot(df, aes(x, y)) +
  geom_hdr_rug_fun(fun_x = dexp, fun_y = dexp) +
  geom_point(size = .5) +
  coord_fixed()

# without data/aesthetic mappings
ggplot() +
```

```
  geom_hdr_rug_fun(fun_x = dexp, fun_y = dexp, xlim = c(0, 7), ylim = c(0, 7)) +
  coord_fixed()


# Plotting univariate normal data, estimating mean and sd
df <- data.frame(x = rnorm(1e4, mean = 1, sd = 3))

# estimating parameters
mu_hat <- mean(df$x)
sd_hat <- sd(df$x)

ggplot(df, aes(x)) +
  geom_hdr_rug_fun(fun_x = dnorm, args_x = list(mean = mu_hat, sd = sd_hat)) +
  geom_density()

# Equivalent to `method_norm_1d()` with `geom_hdr_rug()`
ggplot(df, aes(x)) +
  geom_hdr_rug(method = method_norm_1d()) +
  geom_density()
```

`get_hdr` *Computing the highest density regions of a 2D density*

Description

`get_hdr` is used to estimate a 2-dimensional density and compute corresponding HDRs. The estimated density and HDRs are represented in a discrete form as a grid, defined by arguments `rangex`, `rangey`, and `n`. `get_hdr` is used internally by layer functions `stat_hdr()`, `stat_hdr_points()`, `stat_hdr_fun()`, etc.

Usage

```
get_hdr(
  data = NULL,
  method = "kde",
  probs = c(0.99, 0.95, 0.8, 0.5),
  n = 100,
  rangex = NULL,
  rangey = NULL,
  hdr_membership = TRUE,
  fun,
  args = list()
)
```

Arguments

`data`	A data frame with columns `x` and `y`.
`method`	Either a character (`"kde"`, `"mvnorm"`, `"histogram"`, `"freqpoly"`, or `"fun"`) or `method_*()` function. See the "The `method` argument" section below for details.

probs	Probabilities to compute HDRs for.
n	Resolution of grid representing estimated density and HDRs.
rangex, rangey	Range of grid representing estimated density and HDRs, along the x- and y-axes.
hdr_membership	Should HDR membership of data points (data) be computed? Defaults to TRUE, although it is computationally expensive for large data sets.
fun	Optional, a joint probability density function, must be vectorized in its first two arguments. See the "The fun argument" section below for details.
args	Optional, a list of arguments to be provided to fun.

Value

get_hdr returns a list with elements df_est (data.frame), breaks (named numeric), and data (data.frame).

- df_est: the estimated HDRs and density evaluated on the grid defined by rangex, rangey, and n. The column of estimated HDRs (df_est$hdr) is a numeric vector with values from probs. The columns df_est$fhat and df_est$fhat_discretized correspond to the estimated density on the original scale and rescaled to sum to 1, respectively.
- breaks: the heights of the estimated density (df_est$fhat) corresponding to the HDRs specified by probs. Will always have additional element Inf representing the cutoff for the 100% HDR.
- data: the original data provided in the data argument. If hdr_membership is set to TRUE, this includes a column (data$hdr_membership) with the HDR corresponding to each data point.

The method argument

The density estimator used to estimate the HDRs is specified with the method argument. The simplest way to specify an estimator is to provide a character value to method, for example method = "kde" specifies a kernel density estimator. However, this specification is limited to the default behavior of the estimator.

Instead, it is possible to provide a function call, for example: method = method_kde(). In many cases, these functions accept parameters governing the density estimation procedure. Here, method_kde() accepts parameters h and adjust, both related to the kernel's bandwidth. For details, see ?method_kde. Every method of bivariate density estimation implemented has such corresponding method_*() function, each with an associated help page.

Note: geom_hdr() and other layer functions also have method arguments which behave in the same way. For more details on the use and implementation of the method_*() functions, see vignette("method", "ggdensity").

The fun argument

If method is set to "fun", get_hdr() expects a bivariate probability density function to be specified with the fun argument. It is required that fun be a function of at least two arguments (x and y). Beyond these first two arguments, fun can have arbitrarily many arguments; these can be set in get_hdr() as a named list via the args parameter.

Note: get_hdr() requires that fun be vectorized in x and y. For an example of an appropriate choice of fun, see the final example below.

Examples

```
df <- data.frame(x = rnorm(1e3), y = rnorm(1e3))

# Two ways to specify `method`
get_hdr(df, method = "kde")
get_hdr(df, method = method_kde())

## Not run:

# If parenthesis are omitted, `get_hdr()` errors
get_hdr(df, method = method_kde)

## End(Not run)

# Estimate different HDRs with `probs`
get_hdr(df, method = method_kde(), probs = c(.975, .6, .2))

# Adjust estimator parameters with arguments to `method_kde()`
get_hdr(df, method = method_kde(h = 1))

# Parametric normal estimator of density
get_hdr(df, method = "mvnorm")
get_hdr(df, method = method_mvnorm())

# Compute "population" HDRs of specified bivariate pdf with `method = "fun"`
f <- function(x, y, sd_x = 1, sd_y = 1) dnorm(x, sd = sd_x) * dnorm(y, sd = sd_y)

get_hdr(
  method = "fun", fun = f,
  rangex = c(-5, 5), rangey = c(-5, 5)
 )

get_hdr(
  method = "fun", fun = f,
  rangex = c(-5, 5), rangey = c(-5, 5),
  args = list(sd_x = .5, sd_y = .5) # specify additional arguments w/ `args`
)
```

`get_hdr_1d` *Computing the highest density regions of a 1D density*

Description

`get_hdr_1d` is used to estimate a 1-dimensional density and compute corresponding HDRs. The estimated density and HDRs are represented in a discrete form as a grid, defined by arguments `range` and `n`. `get_hdr_1d` is used internally by layer functions `stat_hdr_rug()` and `stat_hdr_rug_fun()`.

Usage

```
get_hdr_1d(
  x = NULL,
  method = "kde",
  probs = c(0.99, 0.95, 0.8, 0.5),
  n = 512,
  range = NULL,
  hdr_membership = TRUE,
  fun,
  args = list()
)
```

Arguments

x	A vector of data
method	Either a character ("kde", "norm", "histogram", "freqpoly", or "fun") or method_*_1d() function. See the "The method argument" section below for details.
probs	Probabilities to compute HDRs for.
n	Resolution of grid representing estimated density and HDRs.
range	Range of grid representing estimated density and HDRs.
hdr_membership	Should HDR membership of data points (x) be computed?
fun	Optional, a probability density function, must be vectorized in its first argument. See the "The fun argument" section below for details.
args	Optional, a list of arguments to be provided to fun.

Value

get_hdr_1d returns a list with elements df_est (data.frame), breaks (named numeric), and data (data.frame).

- df_est: the estimated HDRs and density evaluated on the grid defined by range and n. The column of estimated HDRs (df_est$hdr) is a numeric vector with values from probs. The columns df_est$fhat and df_est$fhat_discretized correspond to the estimated density on the original scale and rescaled to sum to 1, respectively.
- breaks: the heights of the estimated density (df_est$fhat) corresponding to the HDRs specified by probs. Will always have additional element Inf representing the cutoff for the 100% HDR.
- data: the original data provided in the data argument. If hdr_membership is set to TRUE, this includes a column (data$hdr_membership) with the HDR corresponding to each data point.

The method argument

The density estimator used to estimate the HDRs is specified with the method argument. The simplest way to specify an estimator is to provide a character value to method, for example method

= "kde" specifies a kernel density estimator. However, this specification is limited to the default behavior of the estimator.

Instead, it is possible to provide a function call, for example: method = method_kde_1d(). This is slightly different from the function calls provided in get_hdr(), note the _1d suffix. In many cases, these functions accept parameters governing the density estimation procedure. Here, method_kde_1d() accepts several parameters related to the choice of kernel. For details, see ?method_kde_1d. Every method of univariate density estimation implemented has such corresponding method_*_1d() function, each with an associated help page.

Note: geom_hdr_rug() and other layer functions also have method arguments which behave in the same way. For more details on the use and implementation of the method_*_1d() functions, see vignette("method", "ggdensity").

The fun argument

If method is set to "fun", get_hdr_1d() expects a univariate probability density function to be specified with the fun argument. It is required that fun be a function of at least one argument (x). Beyond this first argument, fun can have arbitrarily many arguments; these can be set in get_hdr_1d() as a named list via the args parameter.

Note: get_hdr_1d() requires that fun be vectorized in x. For an example of an appropriate choice of fun, see the final example below.

Examples

```
x <- rnorm(1e3)

# Two ways to specify `method`
get_hdr_1d(x, method = "kde")
get_hdr_1d(x, method = method_kde_1d())

## Not run:

# If parenthesis are omitted, `get_hdr_1d()` errors
get_hdr_1d(df, method = method_kde_1d)

# If the `_1d` suffix is omitted, `get_hdr_1d()` errors
get_hdr_1d(x, method = method_kde())

## End(Not run)

# Adjust estimator parameters with arguments to `method_kde_1d()`
get_hdr_1d(x, method = method_kde_1d(kernel = "triangular"))

# Estimate different HDRs with `probs`
get_hdr_1d(x, method = method_kde_1d(), probs = c(.975, .6, .2))

# Compute "population" HDRs of specified univariate pdf with `method = "fun"`
f <- function(x, sd = 1) dnorm(x, sd = sd)
get_hdr_1d(method = "fun", fun = f, range = c(-5, 5))
get_hdr_1d(method = "fun", fun = f, range = c(-5, 5), args = list(sd = .5))
```

ggdensity *ggdensity: Stats and Geoms for Density Estimation with ggplot2*

Description

A package that allows more flexible computations for visualization of density estimates with ggplot2.

See Also

Useful links:

- https://jamesotto852.github.io/ggdensity/
- https://github.com/jamesotto852/ggdensity/

method_freqpoly *Bivariate frequency polygon HDR estimator*

Description

Function used to specify bivariate frequency polygon density estimator for `get_hdr()` and layer functions (e.g. `geom_hdr()`).

Usage

```
method_freqpoly(bins = NULL)
```

Arguments

`bins`	Number of bins along each axis. Either a vector of length 2 or a scalar value which is recycled for both dimensions. Defaults to normal reference rule (Scott, pg 87).

Details

For more details on the use and implementation of the `method_*()` functions, see `vignette("method", "ggdensity")`.

References

Scott, David W. Multivariate Density Estimation (2e), Wiley.

Examples

```
set.seed(1)
df <- data.frame(x = rnorm(1e3), y = rnorm(1e3))

ggplot(df, aes(x, y)) +
  geom_hdr(method = method_freqpoly()) +
  geom_point(size = 1)

# The resolution of the frequency polygon estimator can be set via `bins`
ggplot(df, aes(x, y)) +
  geom_hdr(method = method_freqpoly(bins = c(8, 25))) +
  geom_point(size = 1)

# Can also be used with `get_hdr()` for numerical summary of HDRs
res <- get_hdr(df, method = method_freqpoly())
str(res)
```

---

method_freqpoly_1d *Univariate frequency polygon HDR estimator*

Description

Function used to specify univariate frequency polygon density estimator for `get_hdr_1d()` and layer functions (e.g. `geom_hdr_rug()`).

Usage

```
method_freqpoly_1d(bins = NULL)
```

Arguments

`bins`	Number of bins. Defaults to normal reference rule (Scott, pg 59).

Details

For more details on the use and implementation of the `method_*_1d()` functions, see `vignette("method", "ggdensity")`.

References

Scott, David W. Multivariate Density Estimation (2e), Wiley.

Examples

```
df <- data.frame(x = rnorm(1e3))

# Strip chart to visualize 1-d data
p <- ggplot(df, aes(x)) +
  geom_jitter(aes(y = 0), width = 0, height = 2) +
  scale_y_continuous(name = NULL, breaks = NULL) +
  coord_cartesian(ylim = c(-3, 3))

p

p + geom_hdr_rug(method = method_freqpoly_1d())

# The resolution of the frequency polygon estimator can be set via `bins`
p + geom_hdr_rug(method = method_freqpoly_1d(bins = 100))

# Can also be used with `get_hdr_1d()` for numerical summary of HDRs
res <- get_hdr_1d(df$x, method = method_freqpoly_1d())
str(res)
```

---

method_histogram *Bivariate histogram HDR estimator*

Description

Function used to specify bivariate histogram density estimator for `get_hdr()` and layer functions (e.g. `geom_hdr()`).

Usage

```
method_histogram(bins = NULL, smooth = FALSE, nudgex = "none", nudgey = "none")
```

Arguments

`bins`	Number of bins along each axis. Either a vector of length 2 or a scalar value which is recycled for both dimensions. Defaults to normal reference rule (Scott, pg 87).
`smooth`	If `TRUE`, HDRs are smoothed by the marching squares algorithm.
`nudgex, nudgey`	Horizontal and vertical rules for choosing witness points when `smooth == TRUE`. Accepts character vector: `"left"`, `"none"`, `"right"` (`nudgex`) or `"down"`, `"none"`, `"up"` (`nudgey`).

Details

For more details on the use and implementation of the `method_*()` functions, see `vignette("method", "ggdensity")`.

References

Scott, David W. Multivariate Density Estimation (2e), Wiley.

Examples

```
## Not run:

# Histogram estimators can be useful when data has boundary constraints
set.seed(1)
df <- data.frame(x = rexp(1e3), y = rexp(1e3))

ggplot(df, aes(x, y)) +
  geom_hdr(method = method_histogram()) +
  geom_point(size = 1)

# The resolution of the histogram estimator can be set via `bins`
ggplot(df, aes(x, y)) +
  geom_hdr(method = method_histogram(bins = c(8, 25))) +
  geom_point(size = 1)

# By setting `smooth = TRUE`, we can graphically smooth the "blocky" HDRs
ggplot(df, aes(x, y)) +
  geom_hdr(method = method_histogram(smooth = TRUE)) +
  geom_point(size = 1)

# However, we need to set `nudgex` and `nudgey` to align the HDRs correctly
ggplot(df, aes(x, y)) +
  geom_hdr(method = method_histogram(smooth = TRUE, nudgex = "left", nudgey = "down")) +
  geom_point(size = 1)

# Can also be used with `get_hdr()` for numerical summary of HDRs
res <- get_hdr(df, method = method_histogram())
str(res)

## End(Not run)
```

---

method_histogram_1d *Univariate histogram HDR estimator*

Description

Function used to specify univariate histogram density estimator for `get_hdr_1d()` and layer functions (e.g. `geom_hdr_rug()`).

Usage

```
method_histogram_1d(bins = NULL)
```

Arguments

bins	Number of bins. Defaults to normal reference rule (Scott, pg 59).

Details

For more details on the use and implementation of the method_*_1d() functions, see vignette("method", "ggdensity").

References

Scott, David W. Multivariate Density Estimation (2e), Wiley.

Examples

```
# Histogram estimators can be useful when data has boundary constraints
df <- data.frame(x = rexp(1e3))

# Strip chart to visualize 1-d data
p <- ggplot(df, aes(x)) +
  geom_jitter(aes(y = 0), width = 0, height = 2) +
  scale_y_continuous(name = NULL, breaks = NULL) +
  coord_cartesian(ylim = c(-3, 3))

p

p + geom_hdr_rug(method = method_histogram_1d())

# The resolution of the histogram estimator can be set via `bins`
p + geom_hdr_rug(method = method_histogram_1d(bins = 5))

# Can also be used with `get_hdr_1d()` for numerical summary of HDRs
res <- get_hdr_1d(df$x, method = method_histogram_1d())
str(res)
```

---

method_kde *Bivariate kernel density HDR estimator*

Description

Function used to specify bivariate kernel density estimator for get_hdr() and layer functions (e.g. geom_hdr()).

Usage

```
method_kde(h = NULL, adjust = c(1, 1))
```

Arguments

h	Bandwidth (vector of length two). If NULL, estimated using MASS::bandwidth.nrd().
adjust	A multiplicative bandwidth adjustment to be used if 'h' is 'NULL'. This makes it possible to adjust the bandwidth while still using the a bandwidth estimator. For example, adjust = 1/2 means use half of the default bandwidth.

Details

For more details on the use and implementation of the method_*() functions, see vignette("method", "ggdensity").

Examples

```
set.seed(1)
df <- data.frame(x = rnorm(1e3, sd = 3), y = rnorm(1e3, sd = 3))

ggplot(df, aes(x, y)) +
  geom_hdr(method = method_kde()) +
  geom_point(size = 1)

# The defaults of `method_kde()` are the same as the estimator for `ggplot2::geom_density_2d()`
ggplot(df, aes(x, y)) +
  geom_density_2d_filled() +
  geom_hdr_lines(method = method_kde(), probs = seq(.1, .9, by = .1)) +
  theme(legend.position = "none")

# The bandwidth of the estimator can be set directly with `h` or scaled with `adjust`
ggplot(df, aes(x, y)) +
  geom_hdr(method = method_kde(h = 1)) +
  geom_point(size = 1)

ggplot(df, aes(x, y)) +
  geom_hdr(method = method_kde(adjust = 1/2)) +
  geom_point(size = 1)

# Can also be used with `get_hdr()` for numerical summary of HDRs
res <- get_hdr(df, method = method_kde())
str(res)
```

---

method_kde_1d *Univariate kernel density HDR estimator*

Description

Function used to specify univariate kernel density estimator for get_hdr_1d() and layer functions (e.g. geom_hdr_rug()).

Usage

```
method_kde_1d(
  bw = "nrd0",
  adjust = 1,
  kernel = "gaussian",
  weights = NULL,
  window = kernel
)
```

Arguments

bw	the smoothing bandwidth to be used. The kernels are scaled such that this is the standard deviation of the smoothing kernel. (Note this differs from the reference books cited below.) bw can also be a character string giving a rule to choose the bandwidth. See bw.nrd. The default, "nrd0", has remained the default for historical and compatibility reasons, rather than as a general recommendation, where e.g., "SJ" would rather fit, see also Venables and Ripley (2002). The specified (or computed) value of bw is multiplied by adjust.
adjust	the bandwidth used is actually adjust*bw. This makes it easy to specify values like 'half the default' bandwidth.
kernel, window	a character string giving the smoothing kernel to be used. This must partially match one of "gaussian", "rectangular", "triangular", "epanechnikov", "biweight", "cosine" or "optcosine", with default "gaussian", and may be abbreviated to a unique prefix (single letter). "cosine" is smoother than "optcosine", which is the usual 'cosine' kernel in the literature and almost MSE-efficient. However, "cosine" is the version used by S.
weights	numeric vector of non-negative observation weights, hence of same length as x. The default NULL is equivalent to weights = rep(1/nx, nx) where nx is the length of (the finite entries of) x[]. If na.rm = TRUE and there are NA's in x, they *and* the corresponding weights are removed before computations. In that case, when the original weights have summed to one, they are re-scaled to keep doing so. Note that weights are *not* taken into account for automatic bandwidth rules, i.e., when bw is a string. When the weights are proportional to true counts cn, density(x = rep(x, cn)) may be used instead of weights.

Details

For more details on the use and implementation of the method_*_1d() functions, see vignette("method", "ggdensity").

Examples

```
df <- data.frame(x = rnorm(1e3, sd = 3))
```

```
ggplot(df, aes(x)) +
  geom_hdr_rug(method = method_kde_1d()) +
  geom_density()

# Details of the KDE can be adjusted with arguments to `method_kde_1d()`
ggplot(df, aes(x)) +
  geom_hdr_rug(method = method_kde_1d(adjust = 1/5)) +
  geom_density(adjust = 1/5)

ggplot(df, aes(x)) +
  geom_hdr_rug(method = method_kde_1d(kernel = "triangular")) +
  geom_density(kernel = "triangular")

# Can also be used with `get_hdr_1d()` for numerical summary of HDRs
res <- get_hdr_1d(df$x, method = method_kde_1d())
str(res)
```

`method_mvnorm` *Bivariate parametric normal HDR estimator*

Description

Function used to specify bivariate normal density estimator for `get_hdr()` and layer functions (e.g. `geom_hdr()`).

Usage

```
method_mvnorm()
```

Details

For more details on the use and implementation of the `method_*()` functions, see `vignette("method", "ggdensity")`.

Examples

```
# Normal estimator is useful when an assumption of normality is appropriate
set.seed(1)
df <- data.frame(x = rnorm(1e3), y = rnorm(1e3))

ggplot(df, aes(x, y)) +
  geom_hdr(method = method_mvnorm(), xlim = c(-4, 4), ylim = c(-4, 4)) +
  geom_point(size = 1)

# Can also be used with `get_hdr()` for numerical summary of HDRs
res <- get_hdr(df, method = method_mvnorm())
str(res)
```

`method_norm_1d` *Univariate parametric normal HDR estimator*

Description

Function used to specify univariate normal density estimator for `get_hdr_1d()` and layer functions (e.g. `geom_hdr_rug()`).

Usage

```
method_norm_1d()
```

Details

For more details on the use and implementation of the `method_*_1d()` functions, see `vignette("method", "ggdensity")`.

Examples

```
# Normal estimators are useful when an assumption of normality is appropriate
df <- data.frame(x = rnorm(1e3))

ggplot(df, aes(x)) +
  geom_hdr_rug(method = method_norm_1d()) +
  geom_density()

# Can also be used with `get_hdr_1d()` for numerical summary of HDRs
res <- get_hdr_1d(df$x, method = method_norm_1d())
str(res)
```

Index