Package ‘ggdensity’

July 20, 2022

Title Interpretable Bivariate Density Visualization with 'ggplot2'

Version 0.1.0

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 7.2.1

Depends ggplot2

Imports isoband, MASS, stats, scales

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

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

NeedsCompilation no

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

Maintainer James Otto <`jamesotto852@gmail.com`>

Repository CRAN

Date/Publication 2022-07-20 18:00:02 UTC

R topics documented:

`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 the probs 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),
  bins = NULL,
  n = 100,
  xlim = NULL,
  ylim = NULL,
  nudgex = "none",
  nudgey = "none",
  smooth = FALSE,
  adjust = c(1, 1),
  h = 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
)

stat_hdr_lines(
  mapping = NULL,
  data = NULL,
  geom = "hdr_lines",
  position = "identity",
  ...,
  method = "kde",
  probs = c(0.99, 0.95, 0.8, 0.5),
  bins = NULL,
  n = 100,
  xlim = NULL,
  ylim = NULL,
  nudgex = "none",
  nudgey = "none",
  smooth = FALSE,
  adjust = c(1, 1),
  h = NULL,
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE
)

geom_hdr_lines(
  mapping = NULL,
  data = NULL,
  stat = "hdr_lines",
  position = "identity",
  ...,
  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)`).
`geom`	The geometric object to use display the data
`position`	Position adjustment, either as a string, or the result of a call to a position adjustment function.

`...`	Other arguments passed on to `layer()`. These are often aesthetics, used to set an aesthetic to a fixed value, like `colour = "red"` or `size = 3`. They may also be parameters to the paired geom/stat.
`method`	Density estimator to use, accepts character vector: `"kde"`, `"histogram"`, `"freqpoly"`, or `"mvnorm"`.
`probs`	Probabilities to compute highest density regions for.
`bins`	Number of bins along each axis for histogram and frequency polygon estimators. Either a vector of length 2 or a scalar value which is recycled for both dimensions. Defaults to normal reference rule (Scott, pg 87).
`n`	Resolution of grid used in discrete approximations for kernel density and parametric estimators.
`xlim, ylim`	Range to compute and draw regions. If `NULL`, defaults to range of data.
`nudgex`	Horizontal rule for choosing witness points for smoothed histogram method, accepts character vector: `"left"`, `"none"`, `"right"`.
`nudgey`	Vertical rule for choosing witness points for smoothed histogram method, accepts character vector: `"down"`, `"none"`, `"up"`.
`smooth`	If `TRUE`, HDRs computed by the `"histogram"` method are smoothed.
`adjust`	A multiplicative bandwidth adjustment to be used if `h` is `NULL`.
`h`	Bandwidth for kernel density estimator. If `NULL`, estimated using `MASS::bandwidth.nrd()`
`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()`.
`stat`	The statistical transformation to use on the data for this layer, as a string.

Aesthetics

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

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

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

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

Computed variables

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

References

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

Examples

```
# basic simulated data with bivariate normal data and various methods
# (note: code is commented out in this file to save cran check time)
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


# 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()
if (FALSE) {
  pf + geom_hdr(method = "mvnorm")
  pf + geom_hdr(method = "mvnorm", probs = .90, alpha = .5)
  pf + geom_hdr(method = "histogram")
  pf + geom_hdr(method = "freqpoly")
}


# highest density region boundary lines
p + geom_hdr_lines()
p + geom_hdr_lines(method = "mvnorm")
if (FALSE) {
  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
if (FALSE) {
  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")
}
```

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 the probs mapped to the alpha aesthetic by default.

Usage

```
stat_hdr_fun(
  mapping = NULL,
  data = NULL,
  geom = "hdr_fun",
  position = "identity",
  ...,
  fun,
  args = list(),
  normalized = TRUE,
  probs = c(0.99, 0.95, 0.8, 0.5),
  xlim = NULL,
  ylim = NULL,
  res = 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
)

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

geom_hdr_lines_fun(
  mapping = NULL,
  data = NULL,
  stat = "hdr_lines_fun",
  position = "identity",
  ...,
  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)`).
`geom`	The geometric object to use display the data
`position`	Position adjustment, either as a string, or the result of a call to a position adjustment function.
`...`	Other arguments passed on to `layer()`. These are often aesthetics, used to set an aesthetic to a fixed value, like `colour = "red"` or `size = 3`. They may also be parameters to the paired geom/stat.
`fun`	A function, the joint probability density function, must be vectorized in its first two arguments; see examples.
`args`	List of additional arguments passed on to the function `fun` as a named list.
`normalized`	Is the function normalized (a proper PDF)? If no, set to `FALSE`.
`probs`	Probabilities to compute highest density regions for.
`xlim, ylim`	Optionally, restrict the support of the pdf (`fun`) to this range.
`res`	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.
`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()`.
`stat`	The statistical transformation to use on the data for this layer, as a string.

Aesthetics

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

- x

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

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

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

Computed variables

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

Examples

```
f <- function(x, y) dexp(x) * dexp(y)
ggplot() +
  geom_hdr_fun(fun = f, xlim = c(0, 10), ylim = c(0, 10))


# the hdr 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")

ggplot(data, aes(x, y)) +
  geom_hdr_fun(fun = f, args = list(th = th_hat)) +
  geom_point(size = .25, color = "red") +
  xlim(0, 40) + ylim(c(0, 40))
```

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),
  bins = NULL,
  n = 100,
  xlim = NULL,
  ylim = NULL,
  nudgex = "none",
  nudgey = "none",
  smooth = FALSE,
  adjust = c(1, 1),
  h = NULL,
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE
)

geom_hdr_points(
  mapping = NULL,
  data = NULL,
  stat = "hdr_points",
  position = "identity",
  ...,
  method = "kde",
  probs = c(0.99, 0.95, 0.8, 0.5),
  bins = NULL,
  n = 100,
  xlim = NULL,
  ylim = NULL,
  nudgex = "none",
  nudgey = "none",
  smooth = FALSE,
  adjust = c(1, 1),
  h = NULL,
  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)`).
`geom`	The geometric object to use display the data
`position`	Position adjustment, either as a string, or the result of a call to a position adjustment function.

`...`	Other arguments passed on to `layer()`. These are often aesthetics, used to set an aesthetic to a fixed value, like `colour = "red"` or `size = 3`. They may also be parameters to the paired geom/stat.
`method`	Density estimator to use, accepts character vector: `"kde"`, `"histogram"`, `"freqpoly"`, or `"mvnorm"`.
`probs`	Probabilities to compute highest density regions for.
`bins`	Number of bins along each axis for histogram and frequency polygon estimators. Either a vector of length 2 or a scalar value which is recycled for both dimensions. Defaults to normal reference rule (Scott, pg 87).
`n`	Number of grid points in each direction.
`xlim, ylim`	Range to compute and draw regions. If `NULL`, defaults to range of data.
`nudgex`	Horizontal rule for choosing witness points for smoothed histogram method, accepts character vector: `"left"`, `"none"`, `"right"`.
`nudgey`	Vertical rule for choosing witness points for smoothed histogram method, accepts character vector: `"down"`, `"none"`, `"up"`.
`smooth`	If `TRUE`, HDRs computed by the `"histogram"` method are smoothed.
`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.
`h`	Bandwidth (vector of length two). If `NULL`, estimated using `MASS::bandwidth.nrd()`.
`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()`.
`stat`	The statistical transformation to use on the data for this layer, as a string.

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

```
# basic simulated data with bivariate normal data and various methods
# (note: code is commented out in this file to save cran check time)
df <- data.frame(x = rnorm(500), y = rnorm(500))
p <- ggplot(df, aes(x, y)) + coord_equal()
p + geom_hdr_points()
p + geom_hdr_points(method = "mvnorm")
p + geom_hdr_points(method = "freqpoly")
# p + geom_hdr_points(method = "histogram")

# 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) +
 geom_hdr_points(aes(fill = after_stat(probs)), color = "black", shape = 21, size = 2)
```

---

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 = "hdr_points_fun",
  position = "identity",
  ...,
  fun,
  args = list(),
  normalized = TRUE,
  probs = c(0.99, 0.95, 0.8, 0.5),
  xlim = NULL,
  ylim = NULL,
  res = 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()` 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)`).
`geom`	The geometric object to use display the data
`position`	Position adjustment, either as a string, or the result of a call to a position adjustment function.
`...`	Other arguments passed on to `layer()`. These are often aesthetics, used to set an aesthetic to a fixed value, like `colour = "red"` or `size = 3`. They may also be parameters to the paired geom/stat.
`fun`	A function, the joint probability density function, must be vectorized in its first two arguments; see examples.
`args`	List of additional arguments passed on to the function `fun` as a named list.
`normalized`	Is the function normalized (a proper PDF)? If no, set to `FALSE`.
`probs`	Probabilities to compute highest density regions for.
`xlim, ylim`	Optionally, restrict the support of the pdf (`fun`) to this range.
`res`	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.
`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()`.
`stat`	The statistical transformation to use on the data for this layer, as a string.

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:
f <- function(x, y) dexp(x) * dexp(y)
df <- data.frame(x = rexp(1000), y = rexp(1000))

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 the probs mapped to the alpha aesthetic by default.

Usage

```
stat_hdr_rug(
  mapping = NULL,
  data = NULL,
  geom = "hdr_rug",
  position = "identity",
  ...,
  method = "kde",
  probs = c(0.99, 0.95, 0.8, 0.5),
  xlim = NULL,
  ylim = NULL,
  h = "nrd0",
  adjust = 1,
  kernel = "gaussian",
  bins = NULL,
  n = 512,
  na.rm = FALSE,
  show.legend = TRUE,
  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 = TRUE,
  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)`).
`geom`	The geometric object to use display the data
`position`	Position adjustment, either as a string, or the result of a call to a position adjustment function.
`...`	Other arguments passed on to `layer()`. These are often aesthetics, used to set an aesthetic to a fixed value, like `colour = "red"` or `size = 3`. They may also be parameters to the paired geom/stat.
`method`	Density estimator to use, accepts character vector: `"kde"`, `"histogram"`, `"freqpoly"`, or `"mvnorm"`.
`probs`	Probabilities to compute highest density regions for.

`xlim, ylim`	Range to compute and draw regions. If `NULL`, defaults to range of data.
`h`	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 `stats::bw.nrd()`.
`adjust`	A multiplicate bandwidth adjustment. 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.
`kernel`	Kernel. See list of available kernels in `density()`.
`bins`	Number of bins along each axis for histogram and frequency polygon estimators. Either a vector of length 2 or a scalar value which is recycled for both dimensions. Defaults to normal reference rule (Scott, pg 87).
`n`	Resolution of grid used in discrete approximations for kernel density and parametric estimators.
`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()`.
`stat`	The statistical transformation to use on the data for this layer, as a string.
`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):

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

Computed variables

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

References

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

Examples

```
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()

# Or, plot marginal HDR for univariate data
ggplot(df, aes(x)) +
  geom_density() +
  geom_hdr_rug()

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

# Can 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()
```

`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.

Index