Package 'gghdr'

October 27, 2022

Title Visualisation of Highest Density Regions in 'ggplot2'

Version 0.2.0

Description Provides 'ggplot2' framework for visualising Highest Density Regions (HDR) <doi:10.1080/00031305.1996.10474359>. This work is based on the package 'hdrcde'<https://pkg.robjhyndman.com/hdrcde/> and displays highest density regions in 'ggplot2' for one and two dimensions and univariate densities conditional on one covariate.

License GPL-3

Encoding UTF-8

LazyData true

RoxygenNote 7.2.1

URL https://github.com/Sayani07/gghdr

Imports hdrcde, ggplot2, grid, scales, farver, tibble, digest

Suggests spelling, knitr, testthat (>= 2.1.0), vdiffr, rmarkdown, covr

Depends R (>= 2.10)

VignetteBuilder knitr

BugReports https://github.com/Sayani07/gghdr/issues

Language en-GB

NeedsCompilation no

Author Mitchell O'Hara-Wild [aut] (<https://orcid.org/0000-0001-6729-7695>), Stephen Pearce [aut] (<https://orcid.org/0000-0002-9400-8091>), Ryo Nakagawara [aut], Sayani Gupta [aut, cre] (<https://orcid.org/0000-0003-0643-5358>), Darya Vanichkina [aut] (<https://orcid.org/0000-0002-0406-164X>), Emi Tanaka [aut] (<https://orcid.org/0000-0002-1455-259X>), Thomas Fung [aut] (<https://orcid.org/0000-0003-2601-0728>), Rob Hyndman [aut] (<https://orcid.org/0000-0002-2140-5352>)

Maintainer Sayani Gupta <gupta.sayani@gmail.com>

Repository CRAN

Date/Publication 2022-10-27 15:15:19 UTC

R topics documented:

`gghdr-package` *gghdr: Visualisation of Highest Density Regions in 'ggplot2'*

Description

Provides 'ggplot2' framework for visualising Highest Density Regions (HDR) doi: 10.1080/00031305.1996.10474359. This work is based on the package 'hdrcde'`https://pkg.robjhyndman.com/hdrcde/` and displays highest density regions in 'ggplot2' for one and two dimensions and univariate densities conditional on one covariate.

Author(s)

Maintainer: Sayani Gupta <gupta.sayani@gmail.com> (ORCID)

Authors:

- Mitchell O'Hara-Wild <mail@mitchelloharawild.com> (ORCID)
- Stephen Pearce (ORCID)
- Ryo Nakagawara <ryonakagawara@gmail.com>
- Darya Vanichkina <d.vanichkina@gmail.com> (ORCID)
- Emi Tanaka <dr.emi.tanaka@gmail.com> (ORCID)
- Thomas Fung <thomas.fung.dr@gmail.com> (ORCID)
- Rob Hyndman <Rob.Hyndman@monash.edu> (ORCID)

See Also

Useful links:

- `https://github.com/Sayani07/gghdr`
- Report bugs at `https://github.com/Sayani07/gghdr/issues`

draw_key_hdr_boxplot *Draw legend key*

Description

draw legend key for HDR box plot

Usage

```
draw_key_hdr_boxplot(data, params, size)
```

Arguments

data	data
params	parameters
size	size of legend key

faithful *Example dataframe for gghdr*

Description

Example dataframe for gghdr

Usage

```
data(faithful)
```

Format

This contains the 'Old Faithful Geyser' data used in the examples in README and vignettes.

The variables are as follows:

- eruptions. Eruption time in mins
- waiting. Waiting time to next eruption in mins

geom_hdr_boxplot *Box plot for the highest density region*

Description

Box plot for the highest density region

Usage

```
geom_hdr_boxplot(
  mapping = NULL,
  data = NULL,
  stat = "hdrcde",
  position = "dodge2",
  ...,
  varwidth = FALSE,
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE,
  prob = c(0.5, 0.95, 0.99)
)
```

Arguments

mapping	Set of aesthetic mappings created by aes() or aes_(). If specified and inherit.aes = TRUE (the default), it is combined with the default mapping at the top level of the plot. You must supply mapping if there is no plot mapping.
data	The data to be displayed in this layer. There are three options: If NULL, the default, the data is inherited from the plot data as specified in the call to ggplot(). A data.frame, or other object, will override the plot data. All objects will be fortified to produce a data frame. See fortify() for which variables will be created. A function will be called with a single argument, the plot data. The return value must be a data.frame, and will be used as the layer data. A function can be created from a formula (e.g. ~ head(.x, 10)).
stat	The statistical transformation to use on the data for this layer, as a string.
position	Position adjustment, either as a string, or the result of a call to a position adjustment function.
...	Other arguments passed on to layer(). These are often aesthetics, used to set an aesthetic to a fixed value, like colour = "red" or size = 3. They may also be parameters to the paired geom/stat.
varwidth	width, Default: FALSE
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().
prob	Probability coverage required for HDRs, Default: c(0.5, 0.95, 0.99)

Details

Calculates and plots the box plot of highest density regions.

Value

A [ggplot2::Geom] representing a highest density region box plot geometry which can be added to a [ggplot()] object.

Examples

```
library(ggplot2)

ggplot(faithful, aes(y = eruptions)) +
  geom_hdr_boxplot()

ggplot(faithful, aes(y = eruptions, x= waiting)) +
geom_hdr_boxplot(fill="steelblue")
```

---

geom_hdr_rug *Rug plot for the highest density region*

Description

rug visualization for HDR plot

Usage

```
geom_hdr_rug(
  mapping = NULL,
  data = NULL,
  stat = "hdr",
  position = "identity",
  ...,
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE,
  sides = "bl",
  rug_width = unit(0.03, "npc"),
  prob = c(0.5, 0.95, 0.99)
)
```

Arguments

mapping	Default: NULL
data	data
stat	stat
position	Default: "identity"
...	...
na.rm	Default: FALSE
show.legend	Default: NA
inherit.aes	Default: TRUE
sides	position to put rugs
rug_width	width of rugs, Default: unit(0.03, "npc")
prob	PARAM_DESCRIPTION, Default: c(0.5, 0.95, 0.99)

Value

Calculates and plots the rug plot of highest density regions.

Examples

```
library(ggplot2)

ggplot(faithful, aes(x = waiting, y = eruptions)) +
  geom_point() +
  geom_hdr_rug()
```

---

guide_prob *Probability shade bar guide*

Description

The guide shows the colour scales mapped to different probability coverage required for HDRs

Usage

```
guide_prob(title = waiver(), ...)
```

Arguments

title	A character string or expression indicating a title of guide. If NULL, the title is not shown. By default (waiver()), the name of the scale object or the name specified in labs() is used for the title.
...	Further arguments passed onto either guide_colourbar or guide_legend

Value

A guide object

hdr_bin *Binning highest density regions in one or two dimensions*

Description

Binning highest density regions in one or two dimensions

Usage

```
hdr_bin(x, y = NULL, prob = c(0.5, 0.9, 0.99), ...)
```

Arguments

x	Numeric vector
y	Numeric vector of same length as x.
prob	Probability coverage required for HDRs
...	...

Value

probability coverage for each element of the numeric vectors.

Examples

```
library(ggplot2)

ggplot(data = faithful, aes(x = waiting, y = eruptions)) +
  geom_point(aes(colour = hdr_bin(x = waiting, y = eruptions)))
```

lane2 *Example dataframe for gghdr*

Description

Example dataframe for gghdr

Usage

```
data(lane2)
```

Format

This contains the 'lane2' data from the 'hdrcde' package and is used in the examples in README and vignettes.

The variables are as follows:

- flow. a numeric vector giving the traffic flow in vehicles per lane per hour
- speed. a numeric vector giving he speed in miles per hour

`scale_prob` *Probability colour scales*

Description

This set of scales defines new scales for prob geometries equivalent to the ones already defined by ggplot2. This allows the shade of confidence intervals to work with the legend output.

Usage

```
scale_prob_identity(..., guide = "prob")
```

Arguments

`...` Arguments passed on to `continuous_scale`

`scale_name` The name of the scale that should be used for error messages associated with this scale.

`palette` A palette function that when called with a numeric vector with values between 0 and 1 returns the corresponding output values (e.g., `scales::area_pal()`).

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

`breaks` One of:

- `NULL` for no breaks
- `waiver()` for the default breaks computed by the transformation object
- A numeric vector of positions
- A function that takes the limits as input and returns breaks as output (e.g., a function returned by `scales::extended_breaks()`). Also accepts rlang lambda function notation.

`minor_breaks` One of:

- `NULL` for no minor breaks
- `waiver()` for the default breaks (one minor break between each major break)
- A numeric vector of positions
- A function that given the limits returns a vector of minor breaks. Also accepts rlang lambda function notation.

`n.breaks` An integer guiding the number of major breaks. The algorithm may choose a slightly different number to ensure nice break labels. Will only have an effect if `breaks = waiver()`. Use `NULL` to use the default number of breaks given by the transformation.

`labels` One of:

- `NULL` for no labels
- `waiver()` for the default labels computed by the transformation object
- A character vector giving labels (must be same length as `breaks`)
- A function that takes the breaks as input and returns labels as output. Also accepts rlang lambda function notation.

`limits` One of:

- `NULL` to use the default scale range
- A numeric vector of length two providing limits of the scale. Use `NA` to refer to the existing minimum or maximum
- A function that accepts the existing (automatic) limits and returns new limits. Also accepts rlang lambda function notation. Note that setting limits on positional scales will **remove** data outside of the limits. If the purpose is to zoom, use the limit argument in the coordinate system (see `coord_cartesian()`).

`rescaler` A function used to scale the input values to the range [0, 1]. This is always `scales::rescale()`, except for diverging and n colour gradients (i.e., `scale_colour_gradient2()`, `scale_colour_gradientn()`). The `rescaler` is ignored by position scales, which always use `scales::rescale()`. Also accepts rlang lambda function notation.

`oob` One of:

- Function that handles limits outside of the scale limits (out of bounds). Also accepts rlang lambda function notation.
- The default (`scales::censor()`) replaces out of bounds values with `NA`.
- `scales::squish()` for squishing out of bounds values into range.
- `scales::squish_infinite()` for squishing infinite values into range.

`trans` For continuous scales, the name of a transformation object or the object itself. Built-in transformations include "asn", "atanh", "boxcox", "date", "exp", "hms", "identity", "log", "log10", "log1p", "log2", "logit", "modulus", "probability", "probit", "pseudo_log", "reciprocal", "reverse", "sqrt" and "time".

A transformation object bundles together a transform, its inverse, and methods for generating breaks and labels. Transformation objects are defined in the scales package, and are called `<name>_trans` (e.g., `scales::boxcox_trans()`). You can create your own transformation with `scales::trans_new()`.

`expand` For position scales, a vector of range expansion constants used to add some padding around the data to ensure that they are placed some distance away from the axes. Use the convenience function `expansion()` to generate the values for the `expand` argument. The defaults are to expand the scale by 5% on each side for continuous variables, and by 0.6 units on each side for discrete variables.

	`position` For position scales, The position of the axis. `left` or `right` for y axes, `top` or `bottom` for x axes.
	`super` The super class to use for the constructed scale
`guide`	Type of legend. Use `"colourbar"` for continuous colour bar, or `"legend"` for discrete colour legend.

Value

A ggproto object inheriting from ‘[ggplot2::Scale]‘

`stat_hdr` *Stat for hdr box and rug plot*

Description

calculate components of hdr box and rug plot

Usage

```
stat_hdr(
  mapping = NULL,
  data = NULL,
  geom = "hdr_rug",
  position = "dodge2",
  ...,
  coef = 1.5,
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE
)
```

Arguments

`mapping`	Default: NULL
`data`	Default: NULL
`geom`	PARAM_DESCRIPTION, Default: ’hdr_boxplot’
`position`	PARAM_DESCRIPTION, Default: ’dodge2’
`...`	...
`coef`	Default: 1.5
`na.rm`	Default: FALSE
`show.legend`	Default: NA
`inherit.aes`	Default: TRUE

Value

A [ggplot2::Stat] representing the data transformations with required mappings for plotting HDRs using [geom_hdr_boxplot()] and [geom_hdr_rug()].

`stat_hdrcde` *Stat for hdrcde*

Description

compute highest density regions continuously over some conditioned variable

Usage

```
stat_hdrcde(
  mapping = NULL,
  data = NULL,
  geom = "hdr_boxplot",
  position = "dodge2",
  ...,
  coef = 1.5,
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE
)
```

Arguments

`mapping`	Default: NULL
`data`	Default: NULL
`geom`	PARAM_DESCRIPTION, Default: 'hdr_boxplot'
`position`	PARAM_DESCRIPTION, Default: 'dodge2'
`...`	...
`coef`	Default: 1.5
`na.rm`	Default: FALSE
`show.legend`	Default: NA
`inherit.aes`	Default: TRUE

Value

A [ggplot2::Stat] representing the data transformations with required mappings for plotting conditional HDRs using [geom_hdr_boxplot].

Index