

Package ‘cytofan’

May 8, 2026

Type Package

Title Plot Fan Plots for Cytometry Data using 'ggplot2'

Version 0.1.1

Description An implementation of Fan plots for cytometry data in 'ggplot2'.

For reference see Britton, E.; Fisher, P. & J. Whitley (1998) The Inflation Report Projections: Understanding the Fan Chart

<<https://www.bankofengland.co.uk/quarterly-bulletin/1998/q1/the-inflation-report-projections-understanding-the-fan-chart>>).

License GPL-3

Depends R (>= 3.0.0), ggplot2 (>= 2.2.0)

Imports RColorBrewer

Suggests dplyr, reshape2, bodenmiller, knitr, rmarkdown

URL <https://github.com/yannabraham/cytofan>

BugReports <https://github.com/yannabraham/cytofan/issues>

Encoding UTF-8

RoxygenNote 7.3.2

VignetteBuilder knitr

NeedsCompilation no

Author Yann Abraham [aut, cre]

Maintainer Yann Abraham <yann.abraham@gmail.com>

Repository CRAN

Date/Publication 2025-04-08 12:40:02 UTC

Contents

| | |
|----------|---|
| cytofan | 2 |
| do_fan | 2 |
| geom_fan | 3 |
| StatFan | 6 |

| | |
|--------------|----------|
| Index | 7 |
|--------------|----------|

cytofan

cytofan: An implementation of Fan plots in ggplot2.

Description

For reference see [Britton, E.; Fisher, P. & J. Whitley \(1998\) The Inflation Report Projections: Understanding the Fan Chart.](#)

Author(s)

Maintainer: Yann Abraham <yann.abraham@gmail.com>

See Also

Useful links:

- <https://github.com/yannabraham/cytofan>
- Report bugs at <https://github.com/yannabraham/cytofan/issues>

do_fan

Compute summary statistics for stat_fan

Description

Extracts the limits of the Ntiles of a distribution for use in the `stat_fan` function

Usage

```
do_fan(x, step = 0.01)
```

Arguments

| | |
|------|---|
| x | the value to summarize |
| step | the number of bins to break the data into, based on the quantile function |

Value

a data.frame containing

- ymin : the lower limit of the quantile
- ymax : the upper limit of the quantile
- id : an identifier for the quantile
- percent : the fill color to use in `geom_fan`

Examples

```
FanEuStockMarkets <- lapply(colnames(EuStockMarkets),function(id) {  
  res <- do_fan(EuStockMarkets[,id])  
  res$id <- id  
  return(res)  
})  
FanEuStockMarkets <- do.call(rbind,FanEuStockMarkets)
```

geom_fan

Fan plots for trend and population visualizations

Description

Visualise the distribution of continuous variables by dividing each variables into a fixed number of bins and returning the bin limits. In fan plots ('geom_fan') bins are grouped over all variables and colored after their distance from the center bin, which corresponds to the median. The center bin corresponds to the strongest shade of 'colorbase', while other bins get decreasing shades.

Usage

```
geom_fan(  
  mapping = NULL,  
  data = NULL,  
  position = "identity",  
  na.rm = FALSE,  
  show.legend = NA,  
  inherit.aes = TRUE,  
  step = 0.01,  
  colorbase = "Oranges",  
  ...  
)  
  
stat_fan(  
  mapping = NULL,  
  data = NULL,  
  geom = NULL,  
  position = "identity",  
  na.rm = FALSE,  
  show.legend = NA,  
  inherit.aes = TRUE,  
  step = 0.01,  
  ...  
)
```

Arguments

| | |
|-------------|--|
| mapping | Set of aesthetic mappings created by aes() . If specified and <code>inherit.aes = TRUE</code> (the default), it is combined with the default mapping at the top level of the plot. You must supply <code>mapping</code> if there is no plot mapping. |
| data | <p>The data to be displayed in this layer. There are three options:</p> <p>If <code>NULL</code>, the default, the data is inherited from the plot data as specified in the call to ggplot().</p> <p>A <code>data.frame</code>, or other object, will override the plot data. All objects will be fortified to produce a data frame. See fortify() for which variables will be created.</p> <p>A function will be called with a single argument, the plot data. The return value must be a <code>data.frame</code>, and will be used as the layer data. A function can be created from a formula (e.g. <code>~ head(.x, 10)</code>).</p> |
| position | <p>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 <code>position</code> argument accepts the following:</p> <ul style="list-style-type: none"> • 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 <code>position_</code> 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. |
| na.rm | If <code>FALSE</code> , the default, missing values are removed with a warning. If <code>TRUE</code> , missing values are silently removed. |
| show.legend | logical. Should this layer be included in the legends? <code>NA</code> , the default, includes if any aesthetics are mapped. <code>FALSE</code> never includes, and <code>TRUE</code> always includes. It can also be a named logical vector to finely select the aesthetics to display. |
| inherit.aes | If <code>FALSE</code> , overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn't inherit behaviour from the default plot specification, e.g. borders() . |
| step | the number of quantiles to use to compute bins |
| colorbase | the colors to use to draw the ribbon. defaults to RColorBrewer 'Oranges'. See brewer.pal for details. |
| ... | <p>Other arguments passed on to layer()'s <code>params</code> argument. These arguments broadly fall into one of 4 categories below. Notably, further arguments to the <code>position</code> argument, or aesthetics that are required can <i>not</i> be passed through ... Unknown arguments that are not part of the 4 categories below are ignored.</p> <ul style="list-style-type: none"> • 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, <code>colour = "red"</code> or <code>linewidth = 3</code>. The <code>geom</code>'s documentation has an Aesthetics section that lists the available options. The 'required' aesthetics cannot be passed on to the <code>params</code>. 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.

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.

Details

'stat_fan' is suitable only for continuous y data. Moreover, if you have less than '1/step' points you might need to adjust the 'step' parameter.

Computed variables

ymin the lower limit of the quantile

ymax the upper limit of the quantile

id an identifier for the quantile

percent the fill color to use in `geom_fan`

Examples

```
# reformat dataset from short-wide to tall-skinny
EuStockMarkets_ts <- lapply(colnames(EuStockMarkets),function(id) {
  data.frame(id=id,value=as.numeric(EuStockMarkets[,id]))
})
EuStockMarkets_ts <- do.call('rbind',EuStockMarkets_ts)

# plot the distribution of the different stock markets
ggplot(EuStockMarkets_ts,aes(x=id,y=value))+
  geom_fan()

# Change the step
ggplot(EuStockMarkets_ts,aes(x=id,y=value))+
```

```
geom_fan(step=0.05)

# change the default color
ggplot(EuStockMarkets_ts,aes(x=id,y=value))+
  geom_fan(colorbase='Greens')

# any valid RColorBrewer palette will work
ggplot(EuStockMarkets_ts,aes(x=id,y=value))+
  geom_fan(colorbase='RdYlGn')
```

StatFan

StatFan

Description

StatFan

Index

* datasets

StatFan, 6

aes(), 4

borders(), 4

brewer.pal, 4

cytofan, 2

cytofan-package (cytofan), 2

do_fan, 2

fortify(), 4

geom_fan, 3

ggplot(), 4

key glyphs, 5

layer geom, 5

layer position, 4

layer(), 4, 5

stat_fan (geom_fan), 3

StatFan, 6