Package ‘ggExtra’

August 21, 2023

Title Add Marginal Histograms to 'ggplot2', and More 'ggplot2' Enhancements

Version 0.10.1

Description Collection of functions and layers to enhance 'ggplot2'. The flagship function is 'ggMarginal()', which can be used to add marginal histograms/boxplots/density plots to 'ggplot2' scatterplots.

URL `https://github.com/daattali/ggExtra`,

`https://daattali.com/shiny/ggExtra-ggMarginal-demo/`

BugReports `https://github.com/daattali/ggExtra/issues`

Depends R (>= 3.1.0)

Imports colourpicker (>= 1.0), ggplot2 (>= 2.2.0), grDevices, grid (>= 3.1.3), gtable (>= 0.2.0), miniUI (>= 0.1.1), scales (>= 0.2.0), shiny (>= 0.13.0), shinyjs (>= 0.5.2), utils, R6

Suggests knitr (>= 1.7), rmarkdown, rstudioapi (>= 0.5), testthat, vdiffr, fontquiver, svglite, withr, devtools

License MIT + file LICENSE

SystemRequirements pandoc with https support

VignetteBuilder knitr

RoxygenNote 7.2.3

NeedsCompilation no

Author Dean Attali [aut, cre], Christopher Baker [aut]

Maintainer Dean Attali <`daattali@gmail.com`>

Repository CRAN

Date/Publication 2023-08-21 14:40:02 UTC

R topics documented:

`ggMarginal` *Add marginal density/histogram to ggplot2 scatterplots*

Description

Create a ggplot2 scatterplot with marginal density plots (default) or histograms, or add the marginal plots to an existing scatterplot.

Usage

```
ggMarginal(
  p,
  data,
  x,
  y,
  type = c("density", "histogram", "boxplot", "violin", "densigram"),
  margins = c("both", "x", "y"),
  size = 5,
  ...,
  xparams = list(),
  yparams = list(),
  groupColour = FALSE,
  groupFill = FALSE
)
```

Arguments

`p`	A ggplot2 scatterplot to add marginal plots to. If `p` is not provided, then all of `data`, `x`, and `y` must be provided.
`data`	The data.frame to use for creating the marginal plots. Ignored if `p` is provided.
`x`	The name of the variable along the x axis. Ignored if `p` is provided.
`y`	The name of the variable along the y axis. Ignored if `p` is provided.
`type`	What type of marginal plot to show. One of: [density, histogram, boxplot, violin, densigram] (a "densigram" is when a density plot is overlaid on a histogram).
`margins`	Along which margins to show the plots. One of: [both, x, y].

`size`	Integer describing the relative size of the marginal plots compared to the main plot. A size of 5 means that the main plot is 5x wider and 5x taller than the marginal plots.
`...`	Extra parameters to pass to the marginal plots. Any parameter that `geom_line()`, `geom_histogram()`, `geom_boxplot()`, or `geom_violin()` accepts can be used. For example, `colour = "red"` can be used for any marginal plot type, and `binwidth = 10` can be used for histograms.
`xparams`	List of extra parameters to use only for the marginal plot along the x axis.
`yparams`	List of extra parameters to use only for the marginal plot along the y axis.
`groupColour`	If `TRUE`, the colour (or outline) of the marginal plots will be grouped according to the variable mapped to `colour` in the scatter plot. The variable mapped to `colour` in the scatter plot must be a character or factor variable. See examples below.
`groupFill`	If `TRUE`, the fill of the marginal plots will be grouped according to the variable mapped to `colour` in the scatter plot. The variable mapped to `colour` in the scatter plot must be a character or factor variable. See examples below.

Value

An object of class `ggExtraPlot`. This object can be printed to show the plots or saved using any of the typical image-saving functions (for example, using `png()` or `pdf()`).

Note

The `grid` and `gtable` packages are required for this function.

Since the `size` parameter is used by `ggMarginal`, if you want to pass a size to the marginal plots, you cannot use the `...` parameter. Instead, you must pass `size` to both `xparams` and `yparams`. For example, `ggMarginal(p, size = 2)` will change the size of the main vs marginal plot, while `ggMarginal(p, xparams = list(size=2), yparams = list(size=2))` will make the density plot outline thicker.

See Also

Demo Shiny app

Examples

```
## Not run:
library(ggplot2)

# basic usage
p <- ggplot(mtcars, aes(wt, mpg)) + geom_point()
ggMarginal(p)

# using some parameters
set.seed(30)
df <- data.frame(x = rnorm(500, 50, 10), y = runif(500, 0, 50))
p2 <- ggplot(df, aes(x, y)) + geom_point()
ggMarginal(p2)
ggMarginal(p2, type = "histogram")
ggMarginal(p2, margins = "x")
ggMarginal(p2, size = 2)
ggMarginal(p2, colour = "red")
ggMarginal(p2, colour = "red", xparams = list(colour = "blue", size = 3))
ggMarginal(p2, type = "histogram", bins = 10)

# Using violin plot
ggMarginal(p2, type = "violin")

# Using a "densigram" plot
ggMarginal(p2, type = "densigram")

# specifying the data directly instead of providing a plot
ggMarginal(data = df, x = "x", y = "y")

# more examples showing how the marginal plots are properly aligned even when
# the main plot axis/margins/size/etc are changed
set.seed(30)
df2 <- data.frame(x = c(rnorm(250, 50, 10), rnorm(250, 100, 10)),
                  y = runif(500, 0, 50))
p2 <- ggplot(df2, aes(x, y)) + geom_point()
ggMarginal(p2)

p2 <- p2 + ggtitle("Random data") + theme_bw(30)
ggMarginal(p2)

p3 <- ggplot(df2, aes(log(x), y - 500)) + geom_point()
ggMarginal(p3)

p4 <- p3 + scale_x_continuous(limits = c(2, 6)) + theme_bw(50)
ggMarginal(p4)

# Using groupColour and groupFill
# In order to use either of these arguments, we must map 'colour' in the
# scatter plot to a factor or character variable
p <- ggplot(mtcars, aes(x = wt, y = drat, colour = factor(vs))) +
     geom_point()
ggMarginal(p, groupColour = TRUE)
ggMarginal(p, groupColour = TRUE, groupFill = TRUE)

## End(Not run)
```

`ggMarginalGadget` *ggMarginal gadget*

Description

This gadget and addin allow you to select a ggplot2 plot and interactively use `ggMarginal` to build marginal plots on top of your scatterplot.

Usage

```
ggMarginalGadget(plot)
```

Arguments

`plot`	A ggplot2 scatterplot

Value

An object of class `ggExtraPlot`. This object can be printed to show the marginal plots or saved using any of the typical image-saving functions

Note

To use the RStudio addin, highlight the code for a plot in RStudio and select *ggplot2 Marginal Plots* from the RStudio *Addins* menu. This will embed the marginal plots code into your script. Alternatively, you can call `ggMarginalGadget()` with a ggplot2 plot, and the gadget will return a plot object.

Examples

```
if (interactive()) {
  plot <- ggplot2::ggplot(mtcars, ggplot2::aes(wt, mpg)) + ggplot2::geom_point()
  plot2 <- ggMarginalGadget(plot)
}
```

`plotCount` *Plot count data with ggplot2*

Description

Create a bar plot of count (frequency) data that is stored in a data.frame or table.

Usage

```
plotCount(x, ...)
```

Arguments

`x`	A data.frame or table. See 'Details' for more information.
`...`	Extra parameters to pass to the barplot. Any parameter that `geom_bar()` accepts can be used. For example, `fill = "red"` can be used fto make the bars red.

Details

The argument to this function is expected to be either a data.frame or a table.

If a data.frame is provided, it must have exactly two columns: the first column contains the unique values in the data, and the second column is the corresponding integer frequencies to each value.

If a table is provided, it must have exactly one row: the rownames are the unique values in the data, and the row values are the corresponding integer frequencies to each value.

Value

A ggplot2 object that can be have more layers added onto it.

Examples

```
plotCount(table(infert$education))
df <- data.frame("vehicle" = c("bicycle", "car", "unicycle", "Boeing747"),
                 "NumWheels" = c(2, 4, 1, 16))
plotCount(df) + removeGridX()
```

`removeGrid` *Remove grid lines from ggplot2*

Description

Remove grid lines from a ggplot2 plot, to have a cleaner and simpler plot

Usage

```
removeGrid(x = TRUE, y = TRUE)

removeGridX()

removeGridY()
```

Arguments

x	Whether to remove grid lines from the x axis.
y	Whether to remove grid lines from the y axis.

Details

Minor grid lines are always removed.

`removeGrid` removes the major grid lines from the x and/or y axis (both by default).

`removeGridX` is a shortcut for `removeGrid(x = TRUE, y = FALSE)`

`removeGridY` is a shortcut for `removeGrid(x = FALSE, y = TRUE)`

Value

A ggplot2 layer that can be added to an existing ggplot2 object.

Examples

```
df <- data.frame(x = 1:50, y = 1:50)
p <- ggplot2::ggplot(df, ggplot2::aes(x, y)) + ggplot2::geom_point()
p + removeGrid()
p + removeGrid(y = FALSE)
p + removeGridX()
```

`rotateTextX` *Rotate x axis labels*

Description

Rotate the labels on the x axis to be rotated so that they are vertical, which is often useful when there are many overlapping labels along the x axis.

Usage

```
rotateTextX(angle = 90, hjust = 1, vjust = 0.5)
```

Arguments

`angle`	Angle (in [0, 360])
`hjust`	Horizontal justification (in [0, 1])
`vjust`	Vertical justification (in [0, 1])

Details

This function is quite simple, but it can be useful if you don't have the exact syntax to do this engraved in your head.

Value

A ggplot2 layer that can be added to an existing ggplot2 object.

Examples

```
df <- data.frame(x = paste("Letter", LETTERS, sep = "_"),
                 y = seq_along(LETTERS))
p <- ggplot2::ggplot(df, ggplot2::aes(x, y)) + ggplot2::geom_point()
p + rotateTextX()
```

`runExample` *Run ggExtra example*

Description

Launch a Shiny app that shows a demo of what can be done with `ggExtra::ggMarginal`.

Usage

```
runExample()
```

Details

This example is also available online.

Examples

```
## Only run this example in interactive R sessions
if (interactive()) {
  runExample()
}
```

Index