Package 'ggpubr'

March 14, 2017

Type Package

Title 'ggplot2' Based Publication Ready Plots

Version 0.1.2

Date 2017-03-14

Description 'ggplot2' is an excellent and flexible package for elegant data visualization in R. However the default generated plots requires some formatting before we can send them for publication. Furthermore, to customize a 'ggplot', the syntax is opaque and this raises the level of difficulty for researchers with no advanced R programming skills. 'ggpubr' provides some easy-to-use functions for creating and customizing 'ggplot2'- based publication ready plots.

License GPL-2

LazyData TRUE

Depends R (>= 3.1.0), ggplot2

Imports ggrepel, grid, ggsci, plyr, stats, utils

Suggests grDevices, knitr, RColorBrewer, scales

URL http://www.sthda.com/english/rpkgs/ggpubr

BugReports https://github.com/kassambara/ggpubr/issues

RoxygenNote 6.0.1

Collate 'desc_statby.R' 'diff_express.R' 'geom_exec.R' 'get_palette.R' 'utilities.R' 'ggpar.R' 'ggbarplot.R' 'ggboxplot.R' 'ggdensity.R' 'stat_conf_ellipse.R' 'stat_chull.R' 'ggdotchart.R' 'ggdotplot.R' 'ggecdf.R' 'ggerrorplot.R' 'gghistogram.R' 'ggline.R' 'ggmaplot.R' 'ggpie.R' 'ggqqplot.R' 'ggscatter.R' 'ggstripchart.R' 'ggtext.R' 'ggviolin.R' 'show_line_types.R' 'show_point_shapes.R' 'stat_mean.R' 'theme_pubr.R'

NeedsCompilation no

Author Alboukadel Kassambara [aut, cre]

Maintainer Alboukadel Kassambara <alboukadel.kassambara@gmail.com>

Repository CRAN

Date/Publication 2017-03-14 10:26:21

R topics documented:

desc_statby . . . 2
diff_express . . . 4
geom_exec . . . 5
get_palette . . . 5
ggbarplot . . . 7
ggboxplot . . . 11
ggdensity . . . 13
ggdotchart . . . 15
ggdotplot . . . 17
ggecdf . . . 19
ggerrorplot . . . 21
gghistogram . . . 23
ggline . . . 25
ggmaplot . . . 28
ggpar . . . 30
ggpie . . . 33
ggqqplot . . . 35
ggscatter . . . 37
ggstripchart . . . 40
ggtext . . . 43
ggviolin . . . 45
show_line_types . . . 47
show_point_shapes . . . 48
stat_chull . . . 49
stat_conf_ellipse . . . 50
stat_mean . . . 51
theme_pubr . . . 53

Index **54**

`desc_statby` *Descriptive statistics by groups*

Description

Computes descriptive statistics by groups for a measure variable.

Usage

```
desc_statby(data, measure.var, grps, ci = 0.95)
```

Arguments

`data`	a data frame.
`measure.var`	the name of a column containing the variable to be summarized.
`grps`	a character vector containing grouping variables; e.g.: grps = c("grp1", "grp2")
`ci`	the percent range of the confidence interval (default is 0.95).

Value

A data frame containing descriptive statistics, such as:

- **length**: the number of elements in each group
- **min**: minimum
- **max**: maximum
- **median**: median
- **mean**: mean
- **iqr**: interquartile range
- **mad**: median absolute deviation (see ?MAD)
- **sd**: standard deviation of the mean
- **se**: standard error of the mean
- **ci**: confidence interval of the mean
- **range**: the range = max - min
- **cv**: coefficient of variation, sd/mean
- **var**: variance, sd^2

Examples

```
# Load data
data("ToothGrowth")

# Descriptive statistics
res <- desc_statby(ToothGrowth, measure.var = "len",
   grps = c("dose", "supp"))
head(res[, 1:10])
```

diff_express *Differential gene expression analysis results*

Description

Differential gene expression analysis results obtained from comparing the RNAseq data of two different cell populations using DESeq2

Usage

```
data("diff_express")
```

Format

A data frame with 36028 rows and 5 columns.

`name` gene names

`baseMean` mean expression signal accross all samples

`log2FoldChange` log2 fold change

`padj` Adjusted p-value

`detection_call` a numeric vector specifying whether the genes is expressed (value = 1) or not (value = 0).

Examples

```
data(diff_express)

# Default plot
ggmaplot(diff_express, main = expression("Group 1" %->% "Group 2"),
   fdr = 0.05, fc = 2, size = 0.4,
   palette = c("#B31B21", "#1465AC", "darkgray"),
   genenames = as.vector(diff_express$name),
   legend = "top", top = 20,
   font.label = c("bold", 11),
   font.legend = "bold",
   font.main = "bold",
   ggtheme = ggplot2::theme_minimal())

# Add rectangle around labesl
ggmaplot(diff_express, main = expression("Group 1" %->% "Group 2"),
   fdr = 0.05, fc = 2, size = 0.4,
   palette = c("#B31B21", "#1465AC", "darkgray"),
   genenames = as.vector(diff_express$name),
   legend = "top", top = 20,
   font.label = c("bold", 11), label.rectangle = TRUE,
   font.legend = "bold",
   font.main = "bold",
   ggtheme = ggplot2::theme_minimal())
```

geom_exec *Execute ggplot2 functions*

Description

A helper function used by ggpubr functions to execute any geom_* functions in ggplot2. Useful only when you want to call a geom_* function without carrying about the arguments to put in aes(). Basic users of ggpubr don't need this function.

Usage

```
geom_exec(geomfunc = NULL, data = NULL, position = NULL, ...)
```

Arguments

geomfunc	a ggplot2 function (e.g.: geom_point)
data	a data frame to be used for mapping
position	Position adjustment, either as a string, or the result of a call to a position adjustment function.
...	arguments accepeted by the function

Value

return a plot if geomfunc!=Null or a list(option, mapping) if geomfunc = NULL.

Examples

```
## Not run:
ggplot() + geom_exec(geom_point, data = mtcars,
    x = "mpg", y = "wt", size = "cyl", color = "cyl")

## End(Not run)
```

get_palette *Generate Color Palettes*

Description

Generate a palette of k colors from ggsci palettes, RColorbrewer palettes and custom color palettes. Useful to extend RColorBrewer and ggsci to support more colors.

Usage

```
get_palette(palette = "default", k)
```

Arguments

palette	Color palette. Allowed values include: • **Grey color palettes**: "grey" or "gray"; • **RColorBrewer palettes**, see brewer.pal and details section. Examples of palette names include: "RdBu", "Blues", "Dark2", "Set2", ...; • **Custom color palettes**. For example, palette = c("#00AFBB", "#E7B800", "#FC4E07"); • **ggsci scientific journal palettes**, e.g.: "npg", "aaas", "lancet", "jco", "ucscgb", "uchicago", "simpsons" and "rickandmorty".
k	the number of colors to generate.

Details

RColorBrewer palettes: To display all available color palettes, type this in R:RColorBrewer::display.brewer.all(). Color palette names include:

- **Sequential palettes**, suited to ordered data that progress from low to high. Palette names include: Blues BuGn BuPu GnBu Greens Greys Oranges OrRd PuBu PuBuGn PuRd Purples RdPu Reds YlGn YlGnBu YlOrBr YlOrRd.
- **Diverging palettes**:Gradient colors. Names include: BrBG PiYG PRGn PuOr RdBu RdGy RdYlBu RdYlGn Spectral.
- **Qualitative palettes**: Best suited to representing nominal or categorical data. Names include: Accent, Dark2, Paired, Pastel1, Pastel2, Set1, Set2, Set3.

Value

Returns a vector of color palettes.

Examples

```
data("iris")
iris$Species2 <- factor(rep(c(1:10), each = 15))

# Generate a gradient of 10 colors
ggscatter(iris, x = "Sepal.Length", y = "Petal.Length",
 color = "Species2",
 palette = get_palette(c("#00AFBB", "#E7B800", "#FC4E07"), 10))

# Scatter plot with default color palette
ggscatter(iris, x = "Sepal.Length", y = "Petal.Length",
 color = "Species")

# RColorBrewer color palettes
ggscatter(iris, x = "Sepal.Length", y = "Petal.Length",
 color = "Species", palette = get_palette("Dark2", 3))

# ggsci color palettes
ggscatter(iris, x = "Sepal.Length", y = "Petal.Length",
 color = "Species", palette = get_palette("npg", 3))
```

```
# Custom color palette
ggscatter(iris, x = "Sepal.Length", y = "Petal.Length",
 color = "Species",
 palette = c("#00AFBB", "#E7B800", "#FC4E07"))

# Or use this
ggscatter(iris, x = "Sepal.Length", y = "Petal.Length",
 color = "Species",
 palette = get_palette(c("#00AFBB", "#FC4E07"), 3))
```

---

ggbarplot *Bar plot*

Description

Create a bar plot.

Usage

```
ggbarplot(data, x, y, color = "black", fill = "white", palette = NULL,
  size = NULL, width = NULL, label = FALSE, lab.col = "black",
  lab.size = 4, lab.pos = c("out", "in"), lab.vjust = NULL,
  lab.hjust = NULL, select = NULL, order = NULL, sort.val = c("none",
  "desc", "asc"), sort.by.groups = TRUE, top = Inf, add = "none",
  add.params = list(), error.plot = "errorbar",
  position = position_stack(), ggtheme = theme_classic2(), ...)
```

Arguments

data	a data frame
x, y	x and y variables for drawing.
color, fill	outline and fill colors.
palette	the color palette to be used for coloring or filling by groups. Allowed values include "grey" for grey color palettes; brewer palettes e.g. "RdBu", "Blues", ...; or custom color palette e.g. c("blue", "red"); and scientific journal palettes from ggsci R package, e.g.: "npg", "aaas", "lancet", "jco", "ucscgb", "uchicago", "simpsons" and "rickandmorty".
size	Numeric value (e.g.: size = 1). change the size of points and outlines.
width	plot width.
label	specify whether to add labels on the bar plot. Allowed values are:

- **logical value**: If TRUE, y values is added as labels on the bar plot
- **character vector**: Used as text labels; must be the same length as y.

`lab.col, lab.size` text color and size for labels.

`lab.pos` character specifying the position for labels. Allowed values are "out" (for outside) or "in" (for inside). Ignored when lab.vjust != NULL.

`lab.vjust` numeric, vertical justification of labels. Provide negative value (e.g.: -0.4) to put labels outside the bars or positive value to put labels inside (e.g.: 2).

`lab.hjust` numeric, horizontal justification of labels.

`select` character vector specifying which items to display.

`order` character vector specifying the order of items.

`sort.val` a string specifying whether the value should be sorted. Allowed values are "none" (no sorting), "asc" (for ascending) or "desc" (for descending).

`sort.by.groups` logical value. If TRUE the data are sorted by groups. Used only when sort.val != "none".

`top` a numeric value specifying the number of top elements to be shown.

`add` character vector for adding another plot element (e.g.: dot plot or error bars). Allowed values are one or the combination of: "none", "dotplot", "jitter", "boxplot", "point", "mean", "mean_se", "mean_sd", "mean_ci", "mean_range", "median", "median_iqr", "median_mad", "median_range"; see ?desc_statby for more details.

`add.params` parameters (color, shape, size, fill, linetype) for the argument 'add'; e.g.: add.params = list(color = "red").

`error.plot` plot type used to visualize error. Allowed values are one of c("pointrange", "linerange", "crossbar", "errorbar", "upper_errorbar", "lower_errorbar", "upper_pointrange", "lower_pointrange", "upper_linerange", "lower_linerange"). Default value is "pointrange" or "errorbar". Used only when add != "none" and add contains one "mean_*" or "med_*" where "*" = sd, se, ...

`position` Position adjustment, either as a string, or the result of a call to a position adjustment function.

`ggtheme` function, ggplot2 theme name. Default value is theme_pubr(). Allowed values include ggplot2 official themes: theme_gray(), theme_bw(), theme_minimal(), theme_classic(), theme_void(), ...

`...` other arguments to be passed to be passed to ggpar().

Details

The plot can be easily customized using the function ggpar(). Read ?ggpar for changing:

- main title and axis labels: main, xlab, ylab
- axis limits: xlim, ylim (e.g.: ylim = c(0, 30))
- axis scales: xscale, yscale (e.g.: yscale = "log2")
- color palettes: palette = "Dark2" or palette = c("gray", "blue", "red")
- legend title, labels and position: legend = "right"
- plot orientation : orientation = c("vertical", "horizontal", "reverse")

See Also

ggpar, ggline

Examples

```
# Data
df <- data.frame(dose=c("D0.5", "D1", "D2"),
   len=c(4.2, 10, 29.5))
print(df)

# Basic plot with label outsite
# ++++++++++++++++++++++++++++
ggbarplot(df, x = "dose", y = "len",
  label = TRUE, label.pos = "out")

# Change width
ggbarplot(df, x = "dose", y = "len", width = 0.5)

# Change the plot orientation: horizontal
ggbarplot(df, "dose", "len", orientation = "horiz")

# Change the default order of items
ggbarplot(df, "dose", "len",
   order = c("D2", "D1", "D0.5"))


# Change colors
# ++++++++++++++++++++++++++++

# Change fill and outline color
# add labels inside bars
ggbarplot(df, "dose", "len",
 fill = "steelblue", color = "steelblue",
 label = TRUE, lab.pos = "in", lab.col = "white")

# Change colors by groups: dose
# Use custom color palette
 ggbarplot(df, "dose", "len", color = "dose",
   palette = c("#00AFBB", "#E7B800", "#FC4E07"))

# Change fill and outline colors by groups
 ggbarplot(df, "dose", "len",
   fill = "dose", color = "dose",
   palette = c("#00AFBB", "#E7B800", "#FC4E07"))


# Plot with multiple groups
# +++++++++++++++++++++

# Create some data
df2 <- data.frame(supp=rep(c("VC", "OJ"), each=3),
   dose=rep(c("D0.5", "D1", "D2"),2),
```

```
   len=c(6.8, 15, 33, 4.2, 10, 29.5))
print(df2)

# Plot "len" by "dose" and change color by a second group: "supp"
# Add labels inside bars
ggbarplot(df2, "dose", "len",
  fill = "supp", color = "supp", palette = "Paired",
  label = TRUE, lab.col = "white", lab.pos = "in")

# Change position: Interleaved (dodged) bar plot
ggbarplot(df2, "dose", "len",
  fill = "supp", color = "supp", palette = "Paired",
  label = TRUE,
  position = position_dodge(0.9))

# Add points and errors
# ++++++++++++++++++++++++++

# Data: ToothGrowth data set we'll be used.
df3 <- ToothGrowth
head(df3, 10)

# It can be seen that for each group we have
# different values
ggbarplot(df3, x = "dose", y = "len")

# Visualize the mean of each group
ggbarplot(df3, x = "dose", y = "len",
 add = "mean")

# Add error bars: mean_se
# (other values include: mean_sd, mean_ci, median_iqr, ....)
# Add labels
ggbarplot(df3, x = "dose", y = "len",
 add = "mean_se", label = TRUE, lab.vjust = -1.6)

# Use only "upper_errorbar"
ggbarplot(df3, x = "dose", y = "len",
 add = "mean_se", error.plot = "upper_errorbar")

# Change error.plot to "pointrange"
ggbarplot(df3, x = "dose", y = "len",
 add = "mean_se", error.plot = "pointrange")

# Add jitter points and errors (mean_se)
ggbarplot(df3, x = "dose", y = "len",
 add = c("mean_se", "jitter"))

# Add dot and errors (mean_se)
ggbarplot(df3, x = "dose", y = "len",
 add = c("mean_se", "dotplot"))

# Multiple groups with error bars and jitter point
```

```
ggbarplot(df3, x = "dose", y = "len", color = "supp",
 add = "mean_se", palette = c("#00AFBB", "#E7B800"),
 position = position_dodge())
```

---

ggboxplot *Box plot*

Description

Create a box plot with points. Box plots display a group of numerical data through their quartiles.

Usage

```
ggboxplot(data, x, y, color = "black", fill = "white", palette = NULL,
  linetype = "solid", size = NULL, width = 1, notch = FALSE,
  select = NULL, order = NULL, add = "none", add.params = list(),
  error.plot = "pointrange", ggtheme = theme_classic2(), ...)
```

Arguments

data	a data frame
x, y	x and y variables for drawing.
color, fill	outline and fill colors.
palette	the color palette to be used for coloring or filling by groups. Allowed values include "grey" for grey color palettes; brewer palettes e.g. "RdBu", "Blues", ...; or custom color palette e.g. c("blue", "red"); and scientific journal palettes from ggsci R package, e.g.: "npg", "aaas", "lancet", "jco", "ucscgb", "uchicago", "simpsons" and "rickandmorty".
linetype	line types.
size	Numeric value (e.g.: size = 1). change the size of points and outlines.
width	plot width.
notch	if FALSE (default) make a standard box plot. If TRUE, make a notched box plot. Notches are used to compare groups; if the notches of two boxes do not overlap, this suggests that the medians are significantly different.
select	character vector specifying which items to display.
order	character vector specifying the order of items.
add	character vector for adding another plot element (e.g.: dot plot or error bars). Allowed values are one or the combination of: "none", "dotplot", "jitter", "boxplot", "point", "mean", "mean_se", "mean_sd", "mean_ci", "mean_range", "median", "median_iqr", "median_mad", "median_range"; see ?desc_statby for more details.

add.params	parameters (color, shape, size, fill, linetype) for the argument 'add'; e.g.: add.params = list(color = "red").
error.plot	plot type used to visualize error. Allowed values are one of c("pointrange", "linerange", "crossbar", "errorbar", "upper_errorbar", "lower_errorbar", "upper_pointrange", "lower_pointrange", "upper_linerange", "lower_linerange"). Default value is "pointrange" or "errorbar". Used only when add != "none" and add contains one "mean_*" or "med_*" where "*" = sd, se, ...
ggtheme	function, ggplot2 theme name. Default value is theme_pubr(). Allowed values include ggplot2 official themes: theme_gray(), theme_bw(), theme_minimal(), theme_classic(), theme_void(), ...
...	other arguments to be passed to geom_boxplot including linetype, size, etc. (See ?geom_boxplot).

Details

The plot can be easily customized using the function ggpar(). Read ?ggpar for changing:

- main title and axis labels: main, xlab, ylab
- axis limits: xlim, ylim (e.g.: ylim = c(0, 30))
- axis scales: xscale, yscale (e.g.: yscale = "log2")
- color palettes: palette = "Dark2" or palette = c("gray", "blue", "red")
- legend title, labels and position: legend = "right"
- plot orientation : orientation = c("vertical", "horizontal", "reverse")

Suggestions for the argument "add"

Suggested values are one of c("dotplot", "jitter").

See Also

ggpar, ggviolin, ggdotplot and ggstripchart.

Examples

```
# Load data
data("ToothGrowth")
df <- ToothGrowth

# Basic plot
# ++++++++++++++++++++++++++++
# width: change box plots width
ggboxplot(df, x = "dose", y = "len", width = 0.8)

# Change orientation: horizontal
ggboxplot(df, "dose", "len", orientation = "horizontal")

# Notched box plot
ggboxplot(df, x = "dose", y = "len",
```

```
   notch = TRUE)

# Add dots
# ++++++++++++++++++++++++++
ggboxplot(df, x = "dose", y = "len",
   add = "dotplot")

# Add jitter points and change the shape by groups
ggboxplot(df, x = "dose", y = "len",
   add = "jitter", shape = "dose")


# Select and order items
# ++++++++++++++++++++++++++++++

# Select which items to display: "0.5" and "2"
ggboxplot(df, "dose", "len",
   select = c("0.5", "2"))

# Change the default order of items
ggboxplot(df, "dose", "len",
   order = c("2", "1", "0.5"))


# Change colors
# ++++++++++++++++++++++++++
# Change outline and fill colors
 ggboxplot(df, "dose", "len",
   color = "black", fill = "gray")

# Change outline colors by groups: dose
# Use custom color palette
# Add jitter points and change the shape by groups
 ggboxplot(df, "dose", "len",
    color = "dose", palette =c("#00AFBB", "#E7B800", "#FC4E07"),
    add = "jitter", shape = "dose")

# Change fill color by groups: dose
 ggboxplot(df, "dose", "len",
     fill = "dose", palette = c("#00AFBB", "#E7B800", "#FC4E07"))


# Box plot with multiple groups
# +++++++++++++++++++++
# fill or color box plot by a second group : "supp"
ggboxplot(df, "dose", "len", color = "supp",
 palette = c("#00AFBB", "#E7B800"))
```

---

ggdensity *Density plot*

Description

Create a density plot.

Usage

```
ggdensity(data, x, y = "..density..", color = "black", fill = NA,
  palette = NULL, size = NULL, linetype = "solid", alpha = 0.5,
  add = c("none", "mean", "median"), add.params = list(linetype = "dashed"),
  rug = FALSE, ggtheme = theme_classic2(), ...)
```

Arguments

data	a data frame
x	variable to be drawn.
y	one of "..density.." or "..count..".
color, fill	density line color and fill color.
palette	the color palette to be used for coloring or filling by groups. Allowed values include "grey" for grey color palettes; brewer palettes e.g. "RdBu", "Blues", ...; or custom color palette e.g. c("blue", "red"); and scientific journal palettes from ggsci R package, e.g.: "npg", "aaas", "lancet", "jco", "ucscgb", "uchicago", "simpsons" and "rickandmorty".
size	Numeric value (e.g.: size = 1). change the size of points and outlines.
linetype	line type. See show_line_types.
alpha	numeric value specifying fill color transparency. Value should be in [0, 1], where 0 is full transparency and 1 is no transparency.
add	allowed values are one of "mean" or "median" (for adding mean or median line, respectively).
add.params	parameters (color, size, linetype) for the argument 'add'; e.g.: add.params = list(color = "red").
rug	logical value. If TRUE, add marginal rug.
ggtheme	function, ggplot2 theme name. Default value is theme_pubr(). Allowed values include ggplot2 official themes: theme_gray(), theme_bw(), theme_minimal(), theme_classic(), theme_void(), ...
...	other arguments to be passed to geom_density and ggpar.

Details

The plot can be easily customized using the function ggpar(). Read ?ggpar for changing:

- main title and axis labels: main, xlab, ylab
- axis limits: xlim, ylim (e.g.: ylim = c(0, 30))
- axis scales: xscale, yscale (e.g.: yscale = "log2")
- color palettes: palette = "Dark2" or palette = c("gray", "blue", "red")
- legend title, labels and position: legend = "right"
- plot orientation : orientation = c("vertical", "horizontal", "reverse")

See Also

ggpar

Examples

```
# Create some data format
set.seed(1234)
wdata = data.frame(
   sex = factor(rep(c("F", "M"), each=200)),
   weight = c(rnorm(200, 55), rnorm(200, 58)))

head(wdata, 4)

# Basic density plot
 # Add mean line and marginal rug
ggdensity(wdata, x = "weight", fill = "lightgray",
   add = "mean", rug = TRUE)

# Change outline colors by groups ("sex")
# Use custom palette
ggdensity(wdata, x = "weight",
   add = "mean", rug = TRUE,
   color = "sex", palette = c("#00AFBB", "#E7B800"))


# Change outline and fill colors by groups ("sex")
# Use custom palette
ggdensity(wdata, x = "weight",
   add = "mean", rug = TRUE,
   color = "sex", fill = "sex",
   palette = c("#00AFBB", "#E7B800"))
```

---

ggdotchart *Cleveland's Dot Plots*

Description

Draw a Cleveland dot plot.

Usage

```
ggdotchart(data, x, label, group = NULL, color = "black", palette = NULL,
  shape = 19, size = NULL, sorting = c("descending", "ascending"),
  orientation = c("vertical", "horizontal"), ggtheme = theme_bw(), ...)
```

Arguments

data	a data frame
x	x variable for drawing.
label	the name of the column containing point labels.
group	an optional column name indicating how the elements of x are grouped.
color, size	points color and size.
palette	the color palette to be used for coloring or filling by groups. Allowed values include "grey" for grey color palettes; brewer palettes e.g. "RdBu", "Blues", ...; or custom color palette e.g. c("blue", "red"); and scientific journal palettes from ggsci R package, e.g.: "npg", "aaas", "lancet", "jco", "ucscgb", "uchicago", "simpsons" and "rickandmorty".
shape	point shape. See show_point_shapes.
sorting	a character vector for sorting into ascending or descending order. Allowed values are one of "descending" and "ascending". Partial match are allowed (e.g. sorting = "desc" or "asc"). Default is "descending".
orientation	change the orientation of the plot. Allowed values are one of c("vertical", "horizontal", "reverse"). Partial match is allowed.
ggtheme	function, ggplot2 theme name. Default value is theme_pubr(). Allowed values include ggplot2 official themes: theme_gray(), theme_bw(), theme_minimal(), theme_classic(), theme_void(), ...
...	other arguments to be passed to geom_point and ggpar.

Details

The plot can be easily customized using the function ggpar(). Read ?ggpar for changing:

- main title and axis labels: main, xlab, ylab
- axis limits: xlim, ylim (e.g.: ylim = c(0, 30))
- axis scales: xscale, yscale (e.g.: yscale = "log2")
- color palettes: palette = "Dark2" or palette = c("gray", "blue", "red")
- legend title, labels and position: legend = "right"
- plot orientation : orientation = c("vertical", "horizontal", "reverse")

See Also

ggpar

Examples

```
# Load data
data("mtcars")
df <- mtcars
df$cyl <- as.factor(df$cyl)
df$name <- rownames(df)
head(df[, c("wt", "mpg", "cyl")], 3)
```

```
# Basic plot
ggdotchart(df, x = "mpg", label = "name" )

# Change colors by  group cyl
ggdotchart(df, x = "mpg", label = "name",
   group = "cyl", color = "cyl",
   palette = c('#999999','#E69F00','#56B4E9') )

# Use brewer palette
ggdotchart(df, x = "mpg", label = "name",
   group = "cyl", color = "cyl", palette = "Dark2" )

# Change the orientation
# Sort in ascending order
ggdotchart(df, x = "mpg", label = "name",
   group = "cyl", color = "cyl",
    palette = c("#00AFBB", "#E7B800", "#FC4E07"),
   orientation = "horizontal", sorting = "ascending" )
```

---

ggdotplot *Dot plot*

Description

Create a dot plot.

Usage

```
ggdotplot(data, x, y, color = "black", fill = "lightgray", palette = NULL,
  size = NULL, select = NULL, order = NULL, add = "mean_se",
  add.params = list(), error.plot = "pointrange",
  ggtheme = theme_classic2(), ...)
```

Arguments

data	a data frame
x, y	x and y variables for drawing.
color, fill	outline and fill colors.
palette	the color palette to be used for coloring or filling by groups. Allowed values include "grey" for grey color palettes; brewer palettes e.g. "RdBu", "Blues", ...; or custom color palette e.g. c("blue", "red"); and scientific journal palettes from ggsci R package, e.g.: "npg", "aaas", "lancet", "jco", "ucscgb", "uchicago", "simpsons" and "rickandmorty".
size	Numeric value (e.g.: size = 1). change the size of points and outlines.

select	character vector specifying which items to display.
order	character vector specifying the order of items.
add	character vector for adding another plot element (e.g.: dot plot or error bars). Allowed values are one or the combination of: "none", "dotplot", "jitter", "boxplot", "point", "mean", "mean_se", "mean_sd", "mean_ci", "mean_range", "median", "median_iqr", "median_mad", "median_range"; see ?desc_statby for more details.
add.params	parameters (color, shape, size, fill, linetype) for the argument 'add'; e.g.: add.params = list(color = "red").
error.plot	plot type used to visualize error. Allowed values are one of c("pointrange", "linerange", "crossbar", "errorbar", "upper_errorbar", "lower_errorbar", "upper_pointrange", "lower_pointrange", "upper_linerange", "lower_linerange"). Default value is "pointrange" or "errorbar". Used only when add != "none" and add contains one "mean_*" or "med_*" where "*" = sd, se, ...
ggtheme	function, ggplot2 theme name. Default value is theme_pubr(). Allowed values include ggplot2 official themes: theme_gray(), theme_bw(), theme_minimal(), theme_classic(), theme_void(), ...
...	other arguments to be passed to geom_dotplot.

Details

The plot can be easily customized using the function ggpar(). Read ?ggpar for changing:

- main title and axis labels: main, xlab, ylab
- axis limits: xlim, ylim (e.g.: ylim = c(0, 30))
- axis scales: xscale, yscale (e.g.: yscale = "log2")
- color palettes: palette = "Dark2" or palette = c("gray", "blue", "red")
- legend title, labels and position: legend = "right"
- plot orientation : orientation = c("vertical", "horizontal", "reverse")

See Also

ggpar

Examples

```
# Load data
data("ToothGrowth")
df <- ToothGrowth

# Basic plot with summary statistics : mean_sd
# ++++++++++++++++++++++++++
ggdotplot(df, x = "dose", y = "len",
   add = "mean_sd")

# Change error.plot to "crossbar"
ggdotplot(df, x = "dose", y = "len",
```

```
 add = "mean_sd", add.params = list(width = 0.5),
 error.plot = "crossbar")


# Add box plot
ggdotplot(df, x = "dose", y = "len",
 add = "boxplot")

# Add violin + mean_sd
ggdotplot(df, x = "dose", y = "len",
 add = c("violin", "mean_sd"))


# Change colors
# ++++++++++++++++++++++++++
# Change fill and outline colors by groups: dose
# Use custom color palette
 ggdotplot(df, "dose", "len",
     add = "boxplot",
      color = "dose", fill = "dose",
      palette = c("#00AFBB", "#E7B800", "#FC4E07"))


# Plot with multiple groups
# +++++++++++++++++++++
# Change color by a second group : "supp"
ggdotplot(df, "dose", "len", fill = "supp", color = "supp",
    palette = c("#00AFBB", "#E7B800"))
```

---

ggecdf *Empirical cumulative density function*

Description

Empirical Cumulative Density Function (ECDF).

Usage

```
ggecdf(data, x, color = "black", palette = NULL, size = NULL,
  linetype = "solid", ggtheme = theme_classic2(), ...)
```

Arguments

data	a data frame
x	variable to be drawn.
color	line and point color.

palette	the color palette to be used for coloring or filling by groups. Allowed values include "grey" for grey color palettes; brewer palettes e.g. "RdBu", "Blues", ...; or custom color palette e.g. c("blue", "red"); and scientific journal palettes from ggsci R package, e.g.: "npg", "aaas", "lancet", "jco", "ucscgb", "uchicago", "simpsons" and "rickandmorty".
size	line and point size.
linetype	line type. See show_line_types.
ggtheme	function, ggplot2 theme name. Default value is theme_pubr(). Allowed values include ggplot2 official themes: theme_gray(), theme_bw(), theme_minimal(), theme_classic(), theme_void(), ...
...	other arguments to be passed to stat_ecdf and ggpar.

Details

The plot can be easily customized using the function ggpar(). Read ?ggpar for changing:

- main title and axis labels: main, xlab, ylab
- axis limits: xlim, ylim (e.g.: ylim = c(0, 30))
- axis scales: xscale, yscale (e.g.: yscale = "log2")
- color palettes: palette = "Dark2" or palette = c("gray", "blue", "red")
- legend title, labels and position: legend = "right"
- plot orientation : orientation = c("vertical", "horizontal", "reverse")

See Also

ggpar

Examples

```
# Create some data format
set.seed(1234)
wdata = data.frame(
   sex = factor(rep(c("F", "M"), each=200)),
   weight = c(rnorm(200, 55), rnorm(200, 58)))

head(wdata, 4)

# Basic ECDF plot
ggecdf(wdata, x = "weight")

# Change colors and linetype by groups ("sex")
# Use custom palette
ggecdf(wdata, x = "weight",
   color = "sex", linetype = "sex",
   palette = c("#00AFBB", "#E7B800"))
```

ggerrorplot *Visualizing Error*

Description

Visualizing error.

Usage

```
ggerrorplot(data, x, y, desc_stat = "mean_se", color = "black",
  fill = "white", palette = NULL, size = NULL, width = NULL,
  select = NULL, order = NULL, add = "none", add.params = list(),
  error.plot = "pointrange", position = position_dodge(),
  ggtheme = theme_classic2(), ...)
```

Arguments

data	a data frame
x, y	x and y variables for drawing.
desc_stat	descriptive statistics to be used for visualizing errors. Default value is "mean_se". Allowed values are one of , "mean", "mean_se", "mean_sd", "mean_ci", "mean_range", "median", "median_iqr", "median_mad", "median_range"; see desc_statby for more details.
color, fill	outline and fill colors.
palette	the color palette to be used for coloring or filling by groups. Allowed values include "grey" for grey color palettes; brewer palettes e.g. "RdBu", "Blues", ...; or custom color palette e.g. c("blue", "red"); and scientific journal palettes from ggsci R package, e.g.: "npg", "aaas", "lancet", "jco", "ucscgb", "uchicago", "simpsons" and "rickandmorty".
size	Numeric value (e.g.: size = 1). change the size of points and outlines.
width	plot width.
select	character vector specifying which items to display.
order	character vector specifying the order of items.
add	character vector for adding another plot element (e.g.: dot plot or error bars). Allowed values are one or the combination of: "none", "dotplot", "jitter", "boxplot", "point", "mean", "mean_se", "mean_sd", "mean_ci", "mean_range", "median", "median_iqr", "median_mad", "median_range"; see ?desc_statby for more details.
add.params	parameters (color, shape, size, fill, linetype) for the argument 'add'; e.g.: add.params = list(color = "red").
error.plot	plot type used to visualize error. Allowed values are one of c("pointrange", "linerange", "crossbar", "errorbar", "upper_errorbar", "lower_errorbar", "upper_pointrange", "lower_pointrange", "upper_linerange", "lower_linerange"). Default value is "pointrange" or "errorbar". Used only when add != "none" and add contains one "mean_*" or "med_*" where "*" = sd, se, ...

position	Position adjustment, either as a string, or the result of a call to a position adjustment function.
ggtheme	function, ggplot2 theme name. Default value is theme_pubr(). Allowed values include ggplot2 official themes: theme_gray(), theme_bw(), theme_minimal(), theme_classic(), theme_void(), ...
...	other arguments to be passed to be passed to ggpar().

Details

The plot can be easily customized using the function ggpar(). Read ?ggpar for changing:

- main title and axis labels: main, xlab, ylab
- axis limits: xlim, ylim (e.g.: ylim = c(0, 30))
- axis scales: xscale, yscale (e.g.: yscale = "log2")
- color palettes: palette = "Dark2" or palette = c("gray", "blue", "red")
- legend title, labels and position: legend = "right"
- plot orientation : orientation = c("vertical", "horizontal", "reverse")

See Also

ggpar, ggline

Examples

```
# Data: ToothGrowth data set we'll be used.
df<- ToothGrowth
head(df, 10)

# Plot mean_se
ggerrorplot(df, x = "dose", y = "len")


# Change desc_stat to mean_sd
# (other values include: mean_sd, mean_ci, median_iqr, ....)
# Add labels
ggerrorplot(df, x = "dose", y = "len",
 desc_stat = "mean_sd")

# Change error.plot to "errorbar" and add mean point
# Visualize the mean of each group
ggerrorplot(df, x = "dose", y = "len",
 add = "mean", error.plot = "errorbar")

# Horizontal plot
ggerrorplot(df, x = "dose", y = "len",
 add = "mean", error.plot = "errorbar",
 orientation = "horizontal")
```

```
# Change error.plot to "crossbar"
ggerrorplot(df, x = "dose", y = "len",
 error.plot = "crossbar", width = 0.5)


# Add jitter points and errors (mean_se)
ggerrorplot(df, x = "dose", y = "len",
 add = "jitter")

# Add dot and errors (mean_se)
ggerrorplot(df, x = "dose", y = "len",
 add = "dotplot")

# Multiple groups with error bars and jitter point
ggerrorplot(df, x = "dose", y = "len",
 color = "supp", palette = "Paired",
 error.plot = "pointrange",
 position = position_dodge(0.5))
```

---

gghistogram *Histogram plot*

Description

Create a histogram plot.

Usage

```
gghistogram(data, x, y = "..count..", color = "black", fill = NA,
  palette = NULL, size = NULL, linetype = "solid", alpha = 0.5,
  bins = NULL, add = c("none", "mean", "median"),
  add.params = list(linetype = "dashed"), rug = FALSE,
  add_density = FALSE, ggtheme = theme_classic2(), ...)
```

Arguments

data	a data frame
x	variable to be drawn.
y	one of "..density.." or "..count..".
color, fill	histogram line color and fill color.
palette	the color palette to be used for coloring or filling by groups. Allowed values include "grey" for grey color palettes; brewer palettes e.g. "RdBu", "Blues", ...; or custom color palette e.g. c("blue", "red"); and scientific journal palettes from ggsci R package, e.g.: "npg", "aaas", "lancet", "jco", "ucscgb", "uchicago", "simpsons" and "rickandmorty".

size	Numeric value (e.g.: size = 1). change the size of points and outlines.
linetype	line type. See show_line_types.
alpha	numeric value specifying fill color transparency. Value should be in [0, 1], where 0 is full transparency and 1 is no transparency.
bins	Number of bins. Defaults to 30.
add	allowed values are one of "mean" or "median" (for adding mean or median line, respectively).
add.params	parameters (color, size, linetype) for the argument 'add'; e.g.: add.params = list(color = "red").
rug	logical value. If TRUE, add marginal rug.
add_density	logical value. If TRUE, add density curves.
ggtheme	function, ggplot2 theme name. Default value is theme_pubr(). Allowed values include ggplot2 official themes: theme_gray(), theme_bw(), theme_minimal(), theme_classic(), theme_void(), ...
...	other arguments to be passed to geom_histogram and ggpar.

Details

The plot can be easily customized using the function ggpar(). Read ?ggpar for changing:

- main title and axis labels: main, xlab, ylab
- axis limits: xlim, ylim (e.g.: ylim = c(0, 30))
- axis scales: xscale, yscale (e.g.: yscale = "log2")
- color palettes: palette = "Dark2" or palette = c("gray", "blue", "red")
- legend title, labels and position: legend = "right"
- plot orientation : orientation = c("vertical", "horizontal", "reverse")

See Also

ggpar

Examples

```
# Create some data format
set.seed(1234)
wdata = data.frame(
   sex = factor(rep(c("F", "M"), each=200)),
   weight = c(rnorm(200, 55), rnorm(200, 58)))

head(wdata, 4)

# Basic density plot
# Add mean line and marginal rug
gghistogram(wdata, x = "weight", fill = "lightgray",
   add = "mean", rug = TRUE)
```

```
# Change outline colors by groups ("sex")
# Use custom color palette
gghistogram(wdata, x = "weight",
   add = "mean", rug = TRUE,
   color = "sex", palette = c("#00AFBB", "#E7B800"))

# Change outline and fill colors by groups ("sex")
# Use custom color palette
gghistogram(wdata, x = "weight",
   add = "mean", rug = TRUE,
   color = "sex", fill = "sex",
   palette = c("#00AFBB", "#E7B800"))



# Combine histogram and density plots
gghistogram(wdata, x = "weight",
   add = "mean", rug = TRUE,
   fill = "sex", palette = c("#00AFBB", "#E7B800"),
   add_density = TRUE)
```

---

ggline *Line plot*

Description

Create a line plot.

Usage

```
ggline(data, x, y, group = 1, color = "black", palette = NULL,
  linetype = "solid", plot_type = c("b", "l", "p"), size = 0.5,
  shape = 19, select = NULL, order = NULL, add = "none",
  add.params = list(), error.plot = "errorbar",
  ggtheme = theme_classic2(), ...)
```

Arguments

data	a data frame
x, y	x and y variables for drawing.
group	grouping variable to connect points by line. Allowed values are 1 (for one line, one group) or a character vector specifying the name of the grouping variable (case of multiple lines).
color	line colors.

palette	the color palette to be used for coloring or filling by groups. Allowed values include "grey" for grey color palettes; brewer palettes e.g. "RdBu", "Blues", ...; or custom color palette e.g. c("blue", "red"); and scientific journal palettes from ggsci R package, e.g.: "npg", "aaas", "lancet", "jco", "ucscgb", "uchicago", "simpsons" and "rickandmorty".
linetype	line type.
plot_type	plot type. Allowed values are one of "b" for both line and point; "l" for line only; and "p" for point only. Default is "b".
size	Numeric value (e.g.: size = 1). change the size of points and outlines.
shape	point shapes.
select	character vector specifying which items to display.
order	character vector specifying the order of items.
add	character vector for adding another plot element (e.g.: dot plot or error bars). Allowed values are one or the combination of: "none", "dotplot", "jitter", "boxplot", "point", "mean", "mean_se", "mean_sd", "mean_ci", "mean_range", "median", "median_iqr", "median_mad", "median_range"; see ?desc_statby for more details.
add.params	parameters (color, shape, size, fill, linetype) for the argument 'add'; e.g.: add.params = list(color = "red").
error.plot	plot type used to visualize error. Allowed values are one of c("pointrange", "linerange", "crossbar", "errorbar", "upper_errorbar", "lower_errorbar", "upper_pointrange", "lower_pointrange", "upper_linerange", "lower_linerange"). Default value is "pointrange" or "errorbar". Used only when add != "none" and add contains one "mean_*" or "med_*" where "*" = sd, se, ...
ggtheme	function, ggplot2 theme name. Default value is theme_pubr(). Allowed values include ggplot2 official themes: theme_gray(), theme_bw(), theme_minimal(), theme_classic(), theme_void(), ...
...	other arguments to be passed to geom_dotplot.

Details

The plot can be easily customized using the function ggpar(). Read ?ggpar for changing:

- main title and axis labels: main, xlab, ylab
- axis limits: xlim, ylim (e.g.: ylim = c(0, 30))
- axis scales: xscale, yscale (e.g.: yscale = "log2")
- color palettes: palette = "Dark2" or palette = c("gray", "blue", "red")
- legend title, labels and position: legend = "right"
- plot orientation : orientation = c("vertical", "horizontal", "reverse")

See Also

ggpar, ggbarplot

Examples

```
# Data
df <- data.frame(dose=c("D0.5", "D1", "D2"),
   len=c(4.2, 10, 29.5))
print(df)

# Basic plot
# ++++++++++++++++++++++++++++
ggline(df, x = "dose", y = "len")


# Plot with multiple groups
# ++++++++++++++++++++++

# Create some data
df2 <- data.frame(supp=rep(c("VC", "OJ"), each=3),
   dose=rep(c("D0.5", "D1", "D2"),2),
   len=c(6.8, 15, 33, 4.2, 10, 29.5))
print(df2)

# Plot "len" by "dose" and
# Change line types and point shapes by a second groups: "supp"
ggline(df2, "dose", "len",
  linetype = "supp", shape = "supp")


# Change colors
# ++++++++++++++++++++++

# Change color by group: "supp"
# Use custom color palette
ggline(df2, "dose", "len",
   linetype = "supp", shape = "supp",
   color = "supp", palette = c("#00AFBB", "#E7B800"))


# Add points and errors
# ++++++++++++++++++++++++++

# Data: ToothGrowth data set we'll be used.
df3 <- ToothGrowth
head(df3, 10)

# It can be seen that for each group we have
# different values
ggline(df3, x = "dose", y = "len")

# Visualize the mean of each group
ggline(df3, x = "dose", y = "len",
 add = "mean")

# Add error bars: mean_se
```

```
# (other values include: mean_sd, mean_ci, median_iqr, ....)
# Add labels
ggline(df3, x = "dose", y = "len", add = "mean_se")

# Change error.plot to "pointrange"
ggline(df3, x = "dose", y = "len",
 add = "mean_se", error.plot = "pointrange")

# Add jitter points and errors (mean_se)
ggline(df3, x = "dose", y = "len",
 add = c("mean_se", "jitter"))

# Add dot and errors (mean_se)
ggline(df3, x = "dose", y = "len",
 add = c("mean_se", "dotplot"), color = "steelblue")

# Add violin and errors (mean_se)
ggline(df3, x = "dose", y = "len",
 add = c("mean_se", "violin"), color = "steelblue")

# Multiple groups with error bars
# ++++++++++++++++++++++

ggline(df3, x = "dose", y = "len", color = "supp",
 add = "mean_se", palette = c("#00AFBB", "#E7B800"))

# Add jitter
ggline(df3, x = "dose", y = "len", color = "supp",
 add = c("mean_se", "jitter"), palette = c("#00AFBB", "#E7B800"))

# Add dot plot
ggline(df3, x = "dose", y = "len", color = "supp",
 add = c("mean_se", "dotplot"), palette = c("#00AFBB", "#E7B800"))
```

---

ggmaplot *MA-plot from means and log fold changes*

Description

Make MA-plot which is a scatter plot of log2 fold changes (on the y-axis) versus the mean expression signal (on the x-axis).

Usage

```
ggmaplot(data, fdr = 0.05, fc = 1.5, genenames = NULL,
  detection_call = NULL, size = NULL, font.label = c(12, "plain",
  "black"), label.rectangle = FALSE, palette = c("#B31B21", "#1465AC",
  "darkgray"), top = 15, select.top.method = c("padj", "fc"), main = NULL,
```

```
  xlab = "Log2 mean expression", ylab = "Log2 fold change",
  ggtheme = theme_classic2(), ...)
```

Arguments

data	an object of class DESeqResults, get_diff, DE_Results, matrix or data frame containing the columns baseMean, log2FoldChange, and padj. Rows are genes. • baseMean: the mean expression of genes in the two groups. • log2FoldChange: the log2 fold changes of group 2 compared to group 1 • padj: the adjusted p-value of the used statiscal test.
fdr	Accepted false discovery rate for considering genes as differentially expressed.
fc	the fold change threshold. Only genes with a fold change >= fc and padj <= fdr are considered as significantly differentially expressed.
genenames	a character vector of length nrow(data) specifying gene names corresponding to each row. Used for point labels.
detection_call	a numeric vector with length = nrow(data), specifying if the genes is expressed (value = 1) or not (value = 0). For example detection_call = c(1, 1, 0, 1, 0, 1). Default is NULL. If detection_call column is available in data, it will be used.
size	points size.
font.label	a vector of length 3 indicating respectively the size (e.g.: 14), the style (e.g.: "plain", "bold", "italic", "bold.italic") and the color (e.g.: "red") of point labels. For example *font.label = c(14, "bold", "red")*.
label.rectangle	logical value. If TRUE, add rectangle underneath the text, making it easier to read.
palette	the color palette to be used for coloring or filling by groups. Allowed values include "grey" for grey color palettes; brewer palettes e.g. "RdBu", "Blues", ...; or custom color palette e.g. c("blue", "red"); and scientific journal palettes from ggsci R package, e.g.: "npg", "aaas", "lancet", "jco", "ucscgb", "uchicago", "simpsons" and "rickandmorty".
top	the number of top genes to be shown on the plot. Use top = 0 to hide to gene labels.
select.top.method	methods to be used for selecting top genes. Allowed values include "padj" and "fc" for selecting by adjusted p values or fold changes, respectively.
main	plot main title.
xlab	character vector specifying x axis labels, respectively. Use xlab = FALSE to hide xlab.
ylab	character vector specifying y axis labels. Use ylab = FALSE to hide ylab.
ggtheme	function, ggplot2 theme name. Default value is theme_pubr(). Allowed values include ggplot2 official themes: theme_gray(), theme_bw(), theme_minimal(), theme_classic(), theme_void(), ...
...	other arguments to be passed to ggpar.

Value

returns a ggplot.

Examples

```
data(diff_express)

# Default plot
ggmaplot(diff_express, main = expression("Group 1" %->% "Group 2"),
   fdr = 0.05, fc = 2, size = 0.4,
   palette = c("#B31B21", "#1465AC", "darkgray"),
   genenames = as.vector(diff_express$name),
   legend = "top", top = 20,
   font.label = c("bold", 11),
   font.legend = "bold",
   font.main = "bold",
   ggtheme = ggplot2::theme_minimal())

# Add rectangle around labels
ggmaplot(diff_express, main = expression("Group 1" %->% "Group 2"),
   fdr = 0.05, fc = 2, size = 0.4,
   palette = c("#B31B21", "#1465AC", "darkgray"),
   genenames = as.vector(diff_express$name),
   legend = "top", top = 20,
   font.label = c("bold", 11), label.rectangle = TRUE,
   font.legend = "bold",
   font.main = "bold",
   ggtheme = ggplot2::theme_minimal())
```

---

ggpar *Graphical parameters*

Description

Graphical parameters

Usage

```
ggpar(p, palette = NULL, gradient.cols = NULL, main = NULL,
  submain = NULL, caption = NULL, xlab = NULL, ylab = NULL,
  title = NULL, subtitle = NULL, font.main = NULL, font.submain = NULL,
  font.x = NULL, font.y = NULL, font.caption = NULL, font.title = NULL,
  font.subtitle = NULL, xlim = NULL, ylim = NULL, xscale = c("none",
  "log2", "log10", "sqrt"), yscale = c("none", "log2", "log10", "sqrt"),
  format.scale = FALSE, legend = NULL, legend.title = NULL,
  font.legend = NULL, ticks = TRUE, tickslab = TRUE,
  font.tickslab = NULL, xtickslab.rt = 0, ytickslab.rt = 0,
  xticks.by = NULL, yticks.by = NULL, orientation = c("vertical",
  "horizontal", "reverse"), ggtheme = NULL, ...)
```

Arguments

p	an object of class ggplot or a list of ggplots
palette	the color palette to be used for coloring or filling by groups. Allowed values include "grey" for grey color palettes; brewer palettes e.g. "RdBu", "Blues", ...; or custom color palette e.g. c("blue", "red"); and scientific journal palettes from ggsci R package, e.g.: "npg", "aaas", "lancet", "jco", "ucscgb", "uchicago", "simpsons" and "rickandmorty".
gradient.cols	vector of colors to use for n-colour gradient. Allowed values include brewer and ggsci color palettes.
main, title	plot main title.
submain, subtitle	plot subtitle.
caption	plot caption.
xlab	character vector specifying x axis labels, respectively. Use xlab = FALSE to hide xlab.
ylab	character vector specifying y axis labels. Use ylab = FALSE to hide ylab.
font.main, font.submain, font.caption, font.x, font.y	a vector of length 3 indicating respectively the size (e.g.: 14), the style (e.g.: "plain", "bold", "italic", "bold.italic") and the color (e.g.: "red") of main title, subtitle, caption, xlab and ylab, respectively. For example *font.x = c(14, "bold", "red")*. Use font.x = 14, to change only font size; or use font.x = "bold", to change only font face.
font.title, font.subtitle	alias of font.submain and font.submain, respectively.
xlim, ylim	a numeric vector of length 2, specifying x and y axis limits (minimum and maximum), respectively. e.g.: ylim = c(0, 50).
xscale, yscale	x and y axis scale, respectively. Allowed values are one of c("none", "log2", "log10", "sqrt"); e.g.: yscale="log2".
format.scale	logical value. If TRUE, axis tick mark labels will be formatted when xscale or yscale = "log2" or "log10".
legend	character specifying legend position. Allowed values are one of c("top", "bottom", "left", "right", "none"). To remove the legend use legend = "none". Legend position can be also specified using a numeric vector c(x, y); see details section.
legend.title	legend title.
font.legend	legend text font style; e.g.: font.legend = c(10, "plain", "black").
ticks	logical value. Default is TRUE. If FALSE, hide axis tick marks.
tickslab	logical value. Default is TRUE. If FALSE, hide axis tick labels.
font.tickslab	Font style (size, face, color) for tick labels, e.g.: c(14, "bold", "red").
xtickslab.rt, ytickslab.rt	Rotation angle of x and y axis tick labels, respectively. Default value is 0.

xticks.by, yticks.by
: numeric value controlling x and y axis breaks, respectively. For example, if yticks.by = 5, a tick mark is shown on every 5. Default value is NULL.

orientation
: change the orientation of the plot. Allowed values are one of c("vertical", "horizontal", "reverse"). Partial match is allowed.

ggtheme
: function, ggplot2 theme name. Default value is theme_pubr(). Allowed values include ggplot2 official themes: theme_gray(), theme_bw(), theme_minimal(), theme_classic(), theme_void(), ...

...
: not used

Examples

```
# Load data
data("ToothGrowth")
df <- ToothGrowth

# Basic box plot
# ++++++++++++++++++++++++++++

p <- ggboxplot(df, x = "dose", y = "len")

# Change the plot orientation: horizontal
ggpar(p, orientation = "horiz")


 # Change main title and axis labels
 # ++++++++++++++++++++++++++++

 ggpar(p,
   main = "Plot of length \n by dose",
   xlab = "Dose (mg)", ylab = "Length")

 # Title font styles: 'plain', 'italic', 'bold', 'bold.italic'
 ggpar(p,
   main = "Length by dose",
   font.main = c(14,"bold.italic", "red"),
   font.x = c(14, "bold", "#2E9FDF"),
   font.y = c(14, "bold", "#E7B800"))

 # Hide axis labels
 ggpar(p, xlab = FALSE, ylab = FALSE)


# Change colors
# ++++++++++++++++++++++

# Change outline colors by groups: dose
 p2 <- ggboxplot(df, "dose", "len", color = "dose")
 p2

# Use custom color palette
```

```
ggpar(p2, palette = c("#00AFBB", "#E7B800", "#FC4E07"))

# Use brewer palette
ggpar(p2, palette = "Dark2" )

# Use grey palette
ggpar(p2, palette = "grey")

# Use scientific journal palette from ggsci package
ggpar(p2, palette = "npg") # nature

# Axis ticks, limits, scales
# +++++++++++++++++++++++++

# Axis ticks labels and rotation
ggpar(p,
 font.tickslab = c(14,"bold", "#993333"),
 xtickslab.rt = 45, ytickslab.rt = 45)
# Hide axis ticks and tick labels
ggpar(p, ticks = FALSE, tickslab = FALSE)

# Axis limits
ggpar(p, ylim = c(0, 50))

# Axis scale
ggpar(p, yscale = "log2")

# Format axis scale
ggpar(p, yscale = "log2", format.scale = TRUE)

# Legends
# ++++++++++++++++++
# Change legend position and title
ggpar(p2,
 legend = "right", legend.title = "Dose (mg)",
 font.legend = c(10, "bold", "red"))
```

---

ggpie *Pie chart*

Description

Create a pie chart.

Usage

```
ggpie(data, x, label = NULL, lab.pos = c("out", "in"), lab.adjust = 0,
  lab.font = c(4, "bold", "black"), color = "black", fill = "white",
  palette = NULL, size = NULL, ggtheme = theme_classic2(), ...)
```

Arguments

data	a data frame
x	variable containing values for drawing.
label	variable specifying the label of each slice.
lab.pos	character specifying the position for labels. Allowed values are "out" (for outside) or "in" (for inside).
lab.adjust	numeric value, used to adjust label position when lab.pos = "in". Increase or decrease this value to see the effect.
lab.font	a vector of length 3 indicating respectively the size (e.g.: 14), the style (e.g.: "plain", "bold", "italic", "bold.italic") and the color (e.g.: "red") of label font. For example *lab.font= c(4, "bold", "red")*.
color, fill	outline and fill colors.
palette	the color palette to be used for coloring or filling by groups. Allowed values include "grey" for grey color palettes; brewer palettes e.g. "RdBu", "Blues", ...; or custom color palette e.g. c("blue", "red"); and scientific journal palettes from ggsci R package, e.g.: "npg", "aaas", "lancet", "jco", "ucscgb", "uchicago", "simpsons" and "rickandmorty".
size	Numeric value (e.g.: size = 1). change the size of points and outlines.
ggtheme	function, ggplot2 theme name. Default value is theme_pubr(). Allowed values include ggplot2 official themes: theme_gray(), theme_bw(), theme_minimal(), theme_classic(), theme_void(), ...
...	other arguments to be passed to be passed to ggpar().

Details

The plot can be easily customized using the function ggpar(). Read ?ggpar for changing:

- main title and axis labels: main, xlab, ylab
- axis limits: xlim, ylim (e.g.: ylim = c(0, 30))
- axis scales: xscale, yscale (e.g.: yscale = "log2")
- color palettes: palette = "Dark2" or palette = c("gray", "blue", "red")
- legend title, labels and position: legend = "right"
- plot orientation : orientation = c("vertical", "horizontal", "reverse")

See Also

ggpar, ggline

Examples

```
# Data: Create some data
# +++++++++++++++++++++++++++++++

df <- data.frame(
```

```
 group = c("Male", "Female", "Child"),
  value = c(25, 25, 50))

head(df)


# Basic pie charts
# ++++++++++++++++++++++++++++++++

ggpie(df, "value", label = "group")


# Change color
# ++++++++++++++++++++++++++++++++

# Change fill color by group
# set line color to white
# Use custom color palette
 ggpie(df, "value", label = "group",
      fill = "group", color = "white",
       palette = c("#00AFBB", "#E7B800", "#FC4E07") )


# Change label
# ++++++++++++++++++++++++++++++++

# Show group names and value as labels
labs <- paste0(df$group, " (", df$value, "%)")
ggpie(df, "value", label = labs,
   fill = "group", color = "white",
   palette = c("#00AFBB", "#E7B800", "#FC4E07"))

# Change the position and font color of labels
ggpie(df, "value", label = labs,
   lab.pos = "in", lab.font = "white",
   fill = "group", color = "white",
   palette = c("#00AFBB", "#E7B800", "#FC4E07"))
```

---

ggqqplot *QQ Plots*

Description

Quantile-Quantile plot.

Usage

```
ggqqplot(data, x, color = "black", palette = NULL, size = NULL,
  add = c("qqline", "none"), add.params = list(linetype = "solid"),
  conf.int = TRUE, conf.int.level = 0.95, ggtheme = theme_classic2(), ...)
```

Arguments

data	a data frame
x	variable to be drawn.
color	point color.
palette	the color palette to be used for coloring or filling by groups. Allowed values include "grey" for grey color palettes; brewer palettes e.g. "RdBu", "Blues", ...; or custom color palette e.g. c("blue", "red"); and scientific journal palettes from ggsci R package, e.g.: "npg", "aaas", "lancet", "jco", "ucscgb", "uchicago", "simpsons" and "rickandmorty".
size	point size.
add	character vector. Allowed values are one of "none" and "qqline" (for adding qqline).
add.params	parameters (color, size, linetype) for the argument 'add'; e.g.: add.params = list(color = "red").
conf.int	logical value. If TRUE, confidence interval is added.
conf.int.level	the confidence level. Default value is 0.95.
ggtheme	function, ggplot2 theme name. Default value is theme_pubr(). Allowed values include ggplot2 official themes: theme_gray(), theme_bw(), theme_minimal(), theme_classic(), theme_void(), ...
...	other arguments to be passed to stat_qq and ggpar.

Details

The plot can be easily customized using the function ggpar(). Read ?ggpar for changing:

- main title and axis labels: main, xlab, ylab
- axis limits: xlim, ylim (e.g.: ylim = c(0, 30))
- axis scales: xscale, yscale (e.g.: yscale = "log2")
- color palettes: palette = "Dark2" or palette = c("gray", "blue", "red")
- legend title, labels and position: legend = "right"
- plot orientation : orientation = c("vertical", "horizontal", "reverse")

See Also

ggpar

Examples

```
# Create some data format
set.seed(1234)
wdata = data.frame(
   sex = factor(rep(c("F", "M"), each=200)),
   weight = c(rnorm(200, 55), rnorm(200, 58)))

head(wdata, 4)

# Basic QQ plot
ggqqplot(wdata, x = "weight")

# Change colors and shape by groups ("sex")
# Use custom palette
ggqqplot(wdata, x = "weight",
   color = "sex", palette = c("#00AFBB", "#E7B800"))
```

---

ggscatter *Scatter plot*

Description

Create a scatter plot.

Usage

```
ggscatter(data, x, y, color = "black", fill = "lightgray", palette = NULL,
  shape = 19, size = 2, point = TRUE, rug = FALSE, add = c("none",
  "reg.line", "loess"), add.params = list(), conf.int = FALSE,
  conf.int.level = 0.95, fullrange = FALSE, ellipse = FALSE,
  ellipse.level = 0.95, ellipse.type = "norm", ellipse.alpha = 0.1,
  mean.point = FALSE, mean.point.size = ifelse(is.numeric(size), 2 * size,
  size), star.plot = FALSE, star.plot.lty = 1, star.plot.lwd = NULL,
  label = NULL, font.label = c(12, "plain"), label.select = NULL,
  repel = FALSE, label.rectangle = FALSE, cor.coef = FALSE,
  cor.method = "pearson", cor.coef.coord = c(NULL, NULL),
  cor.coef.size = 12, ggp = NULL, show.legend.text = NA,
  ggtheme = theme_classic2(), ...)
```

Arguments

data	a data frame
x, y	x and y variables for drawing.
color, fill	point colors.

palette	the color palette to be used for coloring or filling by groups. Allowed values include "grey" for grey color palettes; brewer palettes e.g. "RdBu", "Blues", ...; or custom color palette e.g. c("blue", "red"); and scientific journal palettes from ggsci R package, e.g.: "npg", "aaas", "lancet", "jco", "ucscgb", "uchicago", "simpsons" and "rickandmorty".
shape	point shape. See show_point_shapes.
size	Numeric value (e.g.: size = 1). change the size of points and outlines.
point	logical value. If TRUE, show points.
rug	logical value. If TRUE, add marginal rug.
add	allowed values are one of "none", "reg.line" (for adding linear regression line) or "loess" (for adding local regression fitting).
add.params	parameters (color, size, linetype) for the argument 'add'; e.g.: add.params = list(color = "red").
conf.int	logical value. If TRUE, adds confidence interval.
conf.int.level	Level controlling confidence region. Default is 95%. Used only when add != "none" and conf.int = TRUE.
fullrange	should the fit span the full range of the plot, or just the data. Used only when add != "none".
ellipse	logical value. If TRUE, draws ellipses around points.
ellipse.level	the size of the concentration ellipse in normal probability.
ellipse.type	Character specifying frame type. Possible values are 'convex', 'confidence' or types supported by stat_ellipse including one of c("t", "norm", "euclid").
ellipse.alpha	Alpha for ellipse specifying the transparency level of fill color. Use alpha = 0 for no fill color.
mean.point	logical value. If TRUE, group mean points are added to the plot.
mean.point.size	numeric value specifying the size of mean points.
star.plot	logical value. If TRUE, a star plot is generated.
star.plot.lty, star.plot.lwd	line type and line width (size) for star plot, respectively.
label	the name of the column containing point labels. Can be also a character vector with length = nrow(data).
font.label	a vector of length 3 indicating respectively the size (e.g.: 14), the style (e.g.: "plain", "bold", "italic", "bold.italic") and the color (e.g.: "red") of point labels. For example *font.label = c(14, "bold", "red")*. To specify only the size and the style, use font.label = c(14, "plain").
label.select	character vector specifying some labels to show.
repel	a logical value, whether to use ggrepel to avoid overplotting text labels or not.
label.rectangle	logical value. If TRUE, add rectangle underneath the text, making it easier to read.

cor.coef	logical value. If TRUE, correlation coefficient with the p-value will be added to the plot.
cor.method	method for computing correlation coefficient. Allowed values are one of "pearson", "kendall", or "spearman".
cor.coef.coord	numeric vector, of length 2, specifying the x and y coordinates of the correlation coefficient. Default values are NULL.
cor.coef.size	correlation coefficient text font size.
ggp	a ggplot. If not NULL, points are added to an existing plot.
show.legend.text	logical. Should text be included in the legends? NA, the default, includes if any aesthetics are mapped. FALSE never includes, and TRUE always includes.
ggtheme	function, ggplot2 theme name. Default value is theme_pubr(). Allowed values include ggplot2 official themes: theme_gray(), theme_bw(), theme_minimal(), theme_classic(), theme_void(), ...
...	other arguments to be passed to geom_point and ggpar.

Details

The plot can be easily customized using the function ggpar(). Read ?ggpar for changing:

- main title and axis labels: main, xlab, ylab
- axis limits: xlim, ylim (e.g.: ylim = c(0, 30))
- axis scales: xscale, yscale (e.g.: yscale = "log2")
- color palettes: palette = "Dark2" or palette = c("gray", "blue", "red")
- legend title, labels and position: legend = "right"
- plot orientation : orientation = c("vertical", "horizontal", "reverse")

See Also

ggpar

Examples

```
# Load data
data("mtcars")
df <- mtcars
df$cyl <- as.factor(df$cyl)
head(df[, c("wt", "mpg", "cyl")], 3)

# Basic plot
# ++++++++++++++++++++++++++
ggscatter(df, x = "wt", y = "mpg",
   color = "black", shape = 21, size = 3, # Points color, shape and size
   add = "reg.line",  # Add regressin line
   add.params = list(color = "blue", fill = "lightgray"), # Customize reg. line
   conf.int = TRUE, # Add confidence interval
   cor.coef = TRUE # Add correlation coefficient
```

```
   )

# loess method: local regression fitting
ggscatter(df, x = "wt", y = "mpg",
   add = "loess", conf.int = TRUE)


# Control point size by continuous variable values ("qsec")
ggscatter(df, x = "wt", y = "mpg",
   color = "#00AFBB", size = "qsec")


# Change colors
# ++++++++++++++++++++++++++
# Use custom color palette
# Add marginal rug
ggscatter(df, x = "wt", y = "mpg", color = "cyl",
   palette = c("#00AFBB", "#E7B800", "#FC4E07") )




# Add group ellipses and mean points
# Add stars
# ++++++++++++++++++
ggscatter(df, x = "wt", y = "mpg",
   color = "cyl", shape = "cyl",
   palette = c("#00AFBB", "#E7B800", "#FC4E07"),
   ellipse = TRUE, mean.point = TRUE,
   star.plot = TRUE)


# Textual annotation
# +++++++++++++++++
df$name <- rownames(df)
ggscatter(df, x = "wt", y = "mpg",
   color = "cyl", palette = c("#00AFBB", "#E7B800", "#FC4E07"),
   label = "name", repel = TRUE)
```

---

ggstripchart *Stripcharts*

Description

Create a stripchart, also known as one dimensional scatter plots. These plots are suitable compared to box plots when sample sizes are small.

Usage

```
ggstripchart(data, x, y, color = "black", fill = "white", palette = NULL,
  shape = 19, size = NULL, select = NULL, order = NULL,
  add = "mean_se", add.params = list(), error.plot = "pointrange",
  position = position_jitter(0.4), ggtheme = theme_classic2(), ...)
```

Arguments

data	a data frame
x, y	x and y variables for drawing.
color, fill	outline and fill colors.
palette	the color palette to be used for coloring or filling by groups. Allowed values include "grey" for grey color palettes; brewer palettes e.g. "RdBu", "Blues", ...; or custom color palette e.g. c("blue", "red"); and scientific journal palettes from ggsci R package, e.g.: "npg", "aaas", "lancet", "jco", "ucscgb", "uchicago", "simpsons" and "rickandmorty".
shape	point shape
size	Numeric value (e.g.: size = 1). change the size of points and outlines.
select	character vector specifying which items to display.
order	character vector specifying the order of items.
add	character vector for adding another plot element (e.g.: dot plot or error bars). Allowed values are one or the combination of: "none", "dotplot", "jitter", "boxplot", "point", "mean", "mean_se", "mean_sd", "mean_ci", "mean_range", "median", "median_iqr", "median_mad", "median_range"; see ?desc_statby for more details.
add.params	parameters (color, shape, size, fill, linetype) for the argument 'add'; e.g.: add.params = list(color = "red").
error.plot	plot type used to visualize error. Allowed values are one of c("pointrange", "linerange", "crossbar", "errorbar", "upper_errorbar", "lower_errorbar", "upper_pointrange", "lower_pointrange", "upper_linerange", "lower_linerange"). Default value is "pointrange" or "errorbar". Used only when add != "none" and add contains one "mean_*" or "med_*" where "*" = sd, se, ...
position	position adjustment, either as a string, or the result of a call to a position adjustment function. Used to adjust position for multiple groups.
ggtheme	function, ggplot2 theme name. Default value is theme_pubr(). Allowed values include ggplot2 official themes: theme_gray(), theme_bw(), theme_minimal(), theme_classic(), theme_void(), ...
...	other arguments to be passed to geom_jitter.

Details

The plot can be easily customized using the function ggpar(). Read ?ggpar for changing:

- main title and axis labels: main, xlab, ylab
- axis limits: xlim, ylim (e.g.: ylim = c(0, 30))

- axis scales: xscale, yscale (e.g.: yscale = "log2")
- color palettes: palette = "Dark2" or palette = c("gray", "blue", "red")
- legend title, labels and position: legend = "right"
- plot orientation : orientation = c("vertical", "horizontal", "reverse")

See Also

ggpar

Examples

```
# Load data
data("ToothGrowth")
df <- ToothGrowth

# Basic plot with summary statistics: mean_se
# ++++++++++++++++++++++++++++
# Change point shapes by groups: "dose"
ggstripchart(df, x = "dose", y = "len",
   shape = "dose", size = 3,
   add = "mean_se")

# Use mean_sd
# Change error.plot to "crossbar"
ggstripchart(df, x = "dose", y = "len",
   shape = "dose", size = 3,
   add = "mean_sd", add.params = list(width = 0.5),
   error.plot = "crossbar")



# Add summary statistics
# ++++++++++++++++++++++++++

# Add box plot
ggstripchart(df, x = "dose", y = "len",
 shape = "dose", add = "boxplot")

# Add violin + mean_sd
ggstripchart(df, x = "dose", y = "len",
 shape = "dose", add = c("violin", "mean_sd"))


# Change colors
# ++++++++++++++++++++++++++++
# Change colors by groups: dose
# Use custom color palette
 ggstripchart(df, "dose", "len",  shape = "dose",
   color = "dose", palette = c("#00AFBB", "#E7B800", "#FC4E07"),
   add = "mean_sd")
```

```
# Plot with multiple groups
# ++++++++++++++++++++++
# Change shape and color by a second group : "supp"
ggstripchart(df, "dose", "len", shape = "supp",
  color = "supp", palette = c("#00AFBB", "#E7B800"))

# Adjust point position
ggstripchart(df, "dose", "len", shape = "supp",
  color = "supp", palette = c("#00AFBB", "#E7B800"),
  position = position_dodge(0.8) )

# You can also use position_jitterdodge()
# but fill aesthetic is required
ggstripchart(df, "dose", "len",  shape = "supp",
   color = "supp", palette = c("#00AFBB", "#E7B800"),
   position = position_jitterdodge() )

# Add boxplot
ggstripchart(df, "dose", "len", shape = "supp",
 color = "supp", palette = c("#00AFBB", "#E7B800"),
 add = "boxplot", add.params = list(color = "black") )
```

---

ggtext *Text*

Description

Add text to a plot.

Usage

```
ggtext(data, x, y, color = "black", palette = NULL, size = 2,
  point = TRUE, label = NULL, font.label = c(12, "plain"),
  label.select = NULL, repel = FALSE, label.rectangle = FALSE,
  ggp = NULL, ggtheme = theme_classic2(), ...)
```

Arguments

data	a data frame
x, y	x and y variables for drawing.
color	point colors.
palette	the color palette to be used for coloring or filling by groups. Allowed values include "grey" for grey color palettes; brewer palettes e.g. "RdBu", "Blues", ...; or custom color palette e.g. c("blue", "red"); and scientific journal palettes from ggsci R package, e.g.: "npg", "aaas", "lancet", "jco", "ucscgb", "uchicago", "simpsons" and "rickandmorty".

size	Numeric value (e.g.: size = 1). change the size of points and outlines.
point	logical value. If TRUE, show points.
label	the name of the column containing point labels. Can be also a character vector with length = nrow(data).
font.label	a vector of length 3 indicating respectively the size (e.g.: 14), the style (e.g.: "plain", "bold", "italic", "bold.italic") and the color (e.g.: "red") of point labels. For example *font.label = c(14, "bold", "red")*. To specify only the size and the style, use font.label = c(14, "plain").
label.select	character vector specifying some labels to show.
repel	a logical value, whether to use ggrepel to avoid overplotting text labels or not.
label.rectangle	logical value. If TRUE, add rectangle underneath the text, making it easier to read.
ggp	a ggplot. If not NULL, points are added to an existing plot.
ggtheme	function, ggplot2 theme name. Default value is theme_pubr(). Allowed values include ggplot2 official themes: theme_gray(), theme_bw(), theme_minimal(), theme_classic(), theme_void(), ...
...	other arguments to be passed to geom_point and ggpar.

Details

The plot can be easily customized using the function ggpar(). Read ?ggpar for changing:

- main title and axis labels: main, xlab, ylab
- axis limits: xlim, ylim (e.g.: ylim = c(0, 30))
- axis scales: xscale, yscale (e.g.: yscale = "log2")
- color palettes: palette = "Dark2" or palette = c("gray", "blue", "red")
- legend title, labels and position: legend = "right"

See Also

ggpar

Examples

```
# Load data
data("mtcars")
df <- mtcars
df$cyl <- as.factor(df$cyl)
df$name <- rownames(df)
head(df[, c("wt", "mpg", "cyl")], 3)

# Textual annotation
# +++++++++++++++++
ggtext(df, x = "wt", y = "mpg",
   color = "cyl", palette = c("#00AFBB", "#E7B800", "#FC4E07"),
   label = "name", repel = TRUE)
```

```
# Add rectangle around label
ggtext(df, x = "wt", y = "mpg",
   color = "cyl", palette = c("#00AFBB", "#E7B800", "#FC4E07"),
   label = "name", repel = TRUE,  label.rectangle = TRUE)
```

---

ggviolin *Violin plot*

Description

Create a violin plot with error bars. Violin plots are similar to box plots, except that they also show the kernel probability density of the data at different values.

Usage

```
ggviolin(data, x, y, color = "black", fill = "white", palette = NULL,
  linetype = "solid", trim = FALSE, size = NULL, width = 1,
  draw_quantiles = NULL, select = NULL, order = NULL, add = "mean_se",
  add.params = list(), error.plot = "pointrange",
  ggtheme = theme_classic2(), ...)
```

Arguments

data	a data frame
x, y	x and y variables for drawing.
color, fill	outline and fill colors.
palette	the color palette to be used for coloring or filling by groups. Allowed values include "grey" for grey color palettes; brewer palettes e.g. "RdBu", "Blues", ...; or custom color palette e.g. c("blue", "red"); and scientific journal palettes from ggsci R package, e.g.: "npg", "aaas", "lancet", "jco", "ucscgb", "uchicago", "simpsons" and "rickandmorty".
linetype	line types.
trim	If TRUE (default), trim the tails of the violins to the range of the data. If FALSE, don't trim the tails.
size	Numeric value (e.g.: size = 1). change the size of points and outlines.
width	violin width.
draw_quantiles	If not(NULL) (default), draw horizontal lines at the given quantiles of the density estimate.
select	character vector specifying which items to display.
order	character vector specifying the order of items.

add	character vector for adding another plot element (e.g.: dot plot or error bars). Allowed values are one or the combination of: "none", "dotplot", "jitter", "boxplot", "point", "mean", "mean_se", "mean_sd", "mean_ci", "mean_range", "median", "median_iqr", "median_mad", "median_range"; see ?desc_statby for more details.
add.params	parameters (color, shape, size, fill, linetype) for the argument 'add'; e.g.: add.params = list(color = "red").
error.plot	plot type used to visualize error. Allowed values are one of c("pointrange", "linerange", "crossbar", "errorbar", "upper_errorbar", "lower_errorbar", "upper_pointrange", "lower_pointrange", "upper_linerange", "lower_linerange"). Default value is "pointrange" or "errorbar". Used only when add != "none" and add contains one "mean_*" or "med_*" where "*" = sd, se, ...
ggtheme	function, ggplot2 theme name. Default value is theme_pubr(). Allowed values include ggplot2 official themes: theme_gray(), theme_bw(), theme_minimal(), theme_classic(), theme_void(), ...
...	other arguments to be passed to geom_violin.

Details

The plot can be easily customized using the function ggpar(). Read ?ggpar for changing:

- main title and axis labels: main, xlab, ylab
- axis limits: xlim, ylim (e.g.: ylim = c(0, 30))
- axis scales: xscale, yscale (e.g.: yscale = "log2")
- color palettes: palette = "Dark2" or palette = c("gray", "blue", "red")
- legend title, labels and position: legend = "right"
- plot orientation : orientation = c("vertical", "horizontal", "reverse")

See Also

ggpar

Examples

```
# Load data
data("ToothGrowth")
df <- ToothGrowth

# Basic plot
# ++++++++++++++++++++++++++++
ggviolin(df, x = "dose", y = "len")
# Change the plot orientation: horizontal
ggviolin(df, "dose", "len", orientation = "horiz")

# Add summary statistics
# ++++++++++++++++++++++++++++
# Draw quantiles
ggviolin(df, "dose", "len", add = "none",
```

```
   draw_quantiles = 0.5)

# Add box plot
ggviolin(df, x = "dose", y = "len",
 add = "boxplot")

ggviolin(df, x = "dose", y = "len",
 add = "dotplot")

# Add jitter points and
# change point shape by groups ("dose")
ggviolin(df, x = "dose", y = "len",
add = "jitter", shape = "dose")


# Add mean_sd + jittered points
ggviolin(df, x = "dose", y = "len",
 add = c("jitter", "mean_sd"))

# Change error.plot to "crossbar"
ggviolin(df, x = "dose", y = "len",
 add = "mean_sd", error.plot = "crossbar")


# Change colors
# ++++++++++++++++++++++++++++
# Change outline and fill colors
ggviolin(df, "dose", "len",
   color = "black", fill = "gray")

# Change outline colors by groups: dose
# Use custom color palette and add boxplot
ggviolin(df, "dose", "len",  color = "dose",
   palette = c("#00AFBB", "#E7B800", "#FC4E07"),
   add = "boxplot")

# Change fill color by groups: dose
# add boxplot with white fill color
ggviolin(df, "dose", "len", fill = "dose",
   palette = c("#00AFBB", "#E7B800", "#FC4E07"),
   add = "boxplot", add.params = list(fill = "white"))


# Plot with multiple groups
# ++++++++++++++++++++
# fill or color box plot by a second group : "supp"
ggviolin(df, "dose", "len", color = "supp",
 palette = c("#00AFBB", "#E7B800"), add = "boxplot")
```

---

show_line_types *Line types available in R*

Description

Show line types available in R.

Usage

```
show_line_types()
```

Value

a ggplot.

See Also

ggpar and ggline.

Examples

```
show_line_types()+
 theme_minimal()
```

`show_point_shapes` *Point shapes available in R*

Description

Show point shapes available in R.

Usage

```
show_point_shapes()
```

Value

a ggplot.

See Also

ggpar and ggline.

Examples

```
show_point_shapes()+
 theme_minimal()
```

stat_chull *Plot convex hull of a set of points*

Description

Plot convex hull of a set of points.

Usage

```
stat_chull(mapping = NULL, data = NULL, geom = "path",
  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.
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.
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.
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.
...	other arguments passed on to layer. These are often aesthetics, used to set an aesthetic to a fixed value, like color = "red" or size = 3. They may also be parameters to the paired geom/stat.

See Also

ggpar, ggscatter

Examples

```
# Load data
data("mtcars")
df <- mtcars
df$cyl <- as.factor(df$cyl)

# scatter plot with convex hull
ggscatter(df, x = "wt", y = "mpg", color = "cyl")+
 stat_chull(aes(color = cyl))

ggscatter(df, x = "wt", y = "mpg", color = "cyl")+
 stat_chull(aes(color = cyl, fill = cyl), alpha = 0.1, geom = "polygon")
```

---

stat_conf_ellipse *Plot confidence ellipses.*

Description

Plot confidence ellipses around barycenters. The method for computing confidence ellipses has been modified from FactoMineR::coord.ellipse.

Usage

```
stat_conf_ellipse(mapping = NULL, data = NULL, geom = "path",
  position = "identity", na.rm = FALSE, show.legend = NA,
  inherit.aes = TRUE, level = 0.95, npoint = 100, bary = 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.
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.
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.
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.
level	confidence level used to construct the ellipses. By default, 0.95.
npoint	number of points used to draw the ellipses.
bary	logical value. If TRUE, the coordinates of the ellipse around the barycentre of individuals are calculated.
...	other arguments passed on to layer. These are often aesthetics, used to set an aesthetic to a fixed value, like color = "red" or size = 3. They may also be parameters to the paired geom/stat.

See Also

stat_conf_ellipse

Examples

```
# Load data
data("mtcars")
df <- mtcars
df$cyl <- as.factor(df$cyl)

# scatter plot with confidence ellipses
ggscatter(df, x = "wt", y = "mpg", color = "cyl")+
 stat_conf_ellipse(aes(color = cyl))

ggscatter(df, x = "wt", y = "mpg", color = "cyl")+
 stat_conf_ellipse(aes(color = cyl, fill = cyl), alpha = 0.1, geom = "polygon")
```

---

stat_mean *Draw group mean points*

Description

Draw the mean point of each group.

Usage

```
stat_mean(mapping = NULL, data = NULL, geom = "point",
  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.
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.
na.rm	If FALSE (the default), removes missing values with a warning. If TRUE silently removes missing values.
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.
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.
...	other arguments to pass to geom_point.

See Also

stat_conf_ellipse, stat_chull and ggscatter

Examples

```
# Load data
data("mtcars")
df <- mtcars
df$cyl <- as.factor(df$cyl)

# Scatter plot with ellipses and group mean points
ggscatter(df, x = "wt", y = "mpg", color = "cyl", ellipse = TRUE)+
 stat_mean(aes(color = cyl, shape = cyl))
```

theme_pubr *Publication ready theme*

Description

- **theme_pubr()**: Create a publication ready theme
- **labs_pubr()**: Format only plot labels to a publication ready style
- **theme_classic()**: Create a classic theme with axis lines

Usage

```
theme_pubr(base_size = 14, base_family = "")

labs_pubr(base_size = 14, base_family = "")

theme_classic2(base_size = 12, base_family = "")
```

Arguments

base_size	base font size
base_family	base font family

Examples

```
p <- ggplot(mtcars, aes(x = wt, y = mpg)) +
   geom_point(aes(color = gear))

# Default plot
p

# Use theme_pubr()
p + theme_pubr()

# Format labels
p + labs_pubr()
```

Index

aes, *49, 50*, *52*
aes_, *49, 50*, *52*

borders, *49*, *51, 52*
brewer.pal, *6*

desc_statby, 2, *21*
diff_express, 4

fortify, *49, 50*, *52*

geom_density, *14*
geom_exec, 5
geom_histogram, *24*
geom_point, *16*, *39*, *44*, *52*
get_palette, 5
ggbarplot, 7, *26*
ggboxplot, 11
ggdensity, 13
ggdotchart, 15
ggdotplot, *12*, 17
ggecdf, 19
ggerrorplot, 21
gghistogram, 23
ggline, *9*, *22*, 25, *34*, *48*
ggmaplot, 28
ggpar, *9*, *12*, *14–16*, *18*, *20*, *22*, *24*, *26*, *29*, 30, *34*, *36*, *39*, *42*, *44*, *46*, *48, 49*
ggpie, 33
ggplot, *49, 50*, *52*
ggqqplot, 35
ggscatter, 37, *49*, *52*
ggstripchart, *12*, 40
ggtext, 43
ggviolin, *12*, 45

labs_pubr *(*theme_pubr*)*, 53
layer, *49*, *51*

show_line_types, *14*, *20*, *24*, 47
show_point_shapes, *16*, *38*, 48
stat_chull, 49, *52*
stat_conf_ellipse, 50, *51, 52*
stat_ecdf, *20*
stat_ellipse, *38*
stat_mean, 51
stat_qq, *36*

theme_classic2 *(*theme_pubr*)*, 53
theme_pubr, 53