Package 'smplot2'

June 24, 2024

Type Package

Title Creating Standalone and Composite Plots in 'ggplot2' for Publications

Version 0.2.4

Maintainer Seung Hyun (Sam) Min <seung.min@mail.mcgill.ca>

Description Provides functions for creating and annotating a composite plot in 'ggplot2'. Offers background themes and shortcut plotting functions that produce figures that are appropriate for the format of scientific journals. Some methods are described in Min and Zhou (2021) <doi:10.3389/fgene.2021.802894>.

Encoding UTF-8

RoxygenNote 7.3.1

License GPL-2

Depends R (>= 3.4.0)

Imports ggplot2, graphics, ggpubr, cowplot, rlang, tibble, dplyr, grid, utils, pwr, stats, Hmisc, zoo, patchwork

URL https://smin95.github.io/dataviz/

BugReports https://github.com/smin95/smplot2/issues

NeedsCompilation no

Author Seung Hyun (Sam) Min [aut, cre]

Repository CRAN

Date/Publication 2024-06-24 03:50:02 UTC

Contents

sm_add_legend 2
sm_add_point 4
sm_add_text 5
sm_auc 6
sm_auc_all 7
sm_bar 8

sm_bland_altman . . . 9
sm_boxplot . . . 10
sm_ci . . . 12
sm_classic . . . 12
sm_color . . . 13
sm_common_axis . . . 14
sm_common_legend . . . 15
sm_common_title . . . 16
sm_common_xlabel . . . 17
sm_common_ylabel . . . 17
sm_corr_avgErr . . . 18
sm_effsize . . . 19
sm_forest . . . 20
sm_forest_annot . . . 22
sm_hgrid . . . 23
sm_hist . . . 24
sm_hvgrid . . . 25
sm_hvgrid_minor . . . 26
sm_minimal . . . 27
sm_palette . . . 27
sm_panel_label . . . 28
sm_plot_clean . . . 29
sm_pointplot . . . 30
sm_power . . . 32
sm_put_together . . . 32
sm_raincloud . . . 35
sm_slope . . . 37
sm_slope_all . . . 39
sm_slope_mean . . . 40
sm_slope_theme . . . 42
sm_statBlandAlt . . . 43
sm_statCorr . . . 44
sm_stdErr . . . 45
sm_vgrid . . . 46
sm_violin . . . 47

Index **49**

`sm_add_legend` *Adding a common legend on a combined figure*

Description

Adding a common legend on a combined figure

Usage

```
sm_add_legend(
  combined_plot,
  x,
  y,
  sampleplot,
  legend,
  direction = "vertical",
  border = TRUE,
  legend_spacing = 0.5,
  border_color = "black",
  font_size = 12
)
```

Arguments

combined_plot	Combined figure, an output from sm_put_together().
x	Location of the legend along the x-axis of the combined figure. The middle origin is at 0.5.
y	Location of the legend along the y-axis of the combined figure. The middle origin is at 0.5.
sampleplot	A variable containing one sample ggplot2 from which the legend can be derived.
legend	Pre-specified layer of legend created with sm_common_legend().
direction	Direction of the legend: 'horizontal' or 'vertical'.
border	If set TRUE, border around the legend will be created. If set FALSE, the border will be removed.
legend_spacing	Spacing within the legend.
border_color	Color of the legend border
font_size	Text size of the legend

Value

It prints a legend on a a combined plot. It can be used to create a common legend for subplots.

Examples

```
library(ggplot2)
library(smplot2)

ggplot(data = mtcars, mapping = aes(x = drat, y = mpg,
fill = as.factor(cyl))) +
 geom_point(shape = 21, color = 'white',
            size = 3) +
 sm_classic(legends=FALSE) -> p1

ggplot(data = mtcars, mapping = aes(x = drat, y = mpg,
                                    fill = as.factor(cyl))) +
```

```
 geom_point(shape = 21, color = 'white',
            size = 3) +
 sm_hvgrid(legends=FALSE) -> p2

combined_fig <- sm_put_together(list(p1,p2), ncol=2,nrow=1)
sm_add_legend(combined_fig, x = 0.1, y = 0.1, sampleplot = p1)
```

`sm_add_point` *Add a point annotation onto the combined plot*

Description

Add a point annotation onto the combined plot

Usage

```
sm_add_point(x, y, size = 10, shape = 16, color = "black", ...)
```

Arguments

`x`	Location of the point annotation along the x-axis of the combined figure. Default is the middle origin (0.5). Values from 0 to 1.
`y`	Location of the point annotation along the y-axis of the combined figure. Default is the middle origin (0.5). Values from 0 to 1.
`size`	Size of the point
`shape`	Shape of the point. Default is set to circle without border (16).
`color`	Color of the point. Default is set to black.
`...`	Other parameters of point that get passed to geom_point().

Value

Prints a point in the combined plot.

Examples

```
library(ggplot2)
library(smplot2)

ggplot(data = mtcars, mapping = aes(x = drat, y = mpg)) +
geom_point(shape = 21, fill = '#0f993d', color = 'white',
          size = 3) -> p1

ggplot(data = mtcars, mapping = aes(x = drat, y = mpg)) +
 geom_point(shape = 21, fill = '#0f993d', color = 'white', size = 3) +
 sm_hvgrid() -> p2

combined_fig <- sm_put_together(list(p1,p2), ncol=2,nrow=1)
combined_fig + sm_add_point(color='red', size = 10, x = .5, y= .5)
```

sm_add_text *Add a text annotation onto the combined plot*

Description

Add a text annotation onto the combined plot

Usage

```
sm_add_text(
  label,
  x = 0.5,
  y = 0.5,
  angle = 0,
  color = "black",
  fontface = "plain",
  size = 10,
  ...
)
```

Arguments

label	Text label in strings.
x	Location of the text annotation along the x-axis of the combined figure. Default is the middle origin (0.5). Values from 0 to 1.
y	Location of the text annotation along the y-axis of the combined figure. Default is the middle origin (0.5). Values from 0 to 1.
angle	Angle of the text. Default is set to 0 (i.e., horizontal orientation).
color	Color of the text. Default is set to 'black'.
fontface	The default is to set the text as plain This can be changed, to either "plain", "bold", "italic", "bold.italic" .
size	Size of the text annotation
...	Other parameters of the text that will be transferred to the function annotate() from ggplot2.

Value

Prints a text in the combined plot.

Examples

```
library(smplot2)
library(ggplot2)
ggplot(data = mtcars, mapping = aes(x = drat, y = mpg)) +
 geom_point(shape = 21, fill = '#0f993d', color = 'white',
            size = 3) -> p1
```

```
ggplot(data = mtcars, mapping = aes(x = drat, y = mpg)) +
 geom_point(shape = 21, fill = '#0f993d', color = 'white', size = 3) +
 sm_hvgrid() -> p2

combined_fig <- sm_put_together(list(p1,p2), ncol=2,nrow=1)
combined_fig + sm_add_text(label='My label', x = .5, y= .5)
```

---

sm_auc *Calculation of the Area under a Curve (Trapezoidal numerical integration)*

Description

This is equivalent to Matlab's trapz function.

Usage

```
sm_auc(x, y)
```

Arguments

x	This is the scalar spacing of the coordinate. When the argument for 'x' is not provided, it will calculates the approximate integral value for 'Y' with unit spacing based on the length of 'y'.
y	Numerical data. The length of x and y must be equal.

Value

A vector that is the value from the trapezoidal integration is returned.

Examples

```
library(smplot2)
X = c(1,2,3,4,5)
Y1 = c(2,3,4,2,3)
Y2 = c(3,3,3,3,3)

sm_auc(X,Y2)
sm_auc(X,Y1)
```

sm_auc_all *Calculating Area under Curve across multiple conditions and subjects*

Description

This function returns a data frame containing AUCs from a data frame that contains the original raw data. One of the two arguments 'groups' and 'conditions' must be filled. The function will throw an error if both arguments are empty.

Usage

```
sm_auc_all(data, subjects, groups, conditions, x, values)
```

Arguments

data	Name of the variable that stores the data frame that contains the columns with the specified column names.
subjects	The name of the column of the data frame that contains subjects. It must be strings.
groups	The name of the column of the data frame that contains each group. It must be strings.
conditions	The name of the column of the data frame that contains each condition. It must be strings.
x	The name of the column of the data frame that contains the x-axis points/x coordinates from which the AUC can be calculated. It must be strings. The column must not have characters.
values	The name of the column of the data frame that contains the actual data, which are the y-axis points from which the AUC can be calculated. It must be strings.

Value

Returns a data frame containing area under curve from each subject and experimental condition and/or group.

Examples

```
library(smplot2)
set.seed(1) # generate random data
day1 = rnorm(16,0,1)
day2 = rnorm(16,5,1)
Subject <- rep(paste0('S',seq(1:16)), 2)
Value <- data.frame(Value = matrix(c(day1,day2),ncol=1))
Day <- rep(c(1,2), each = length(day1))
Condition <- rep('Control', length(day1)*2)
df <- cbind(Subject, Value, Condition, Day)

sm_auc_all(data = df, subjects = 'Subject',values = 'Value',
```

```
conditions = 'Condition',x = 'Day')
```

---

sm_bar *A bar plot with jittered individual points*

Description

A bar plot with jittered individual points

Usage

```
sm_bar(
  ...,
 bar.params = list(width = 0.7, alpha = 1, color = "transparent", fill = "gray80"),
  err.params = list(linewidth = 1, color = "black"),
  point.params = list(size = 2.5, alpha = 0.65, shape = 16),
  errorbar_type = "se",
  point_jitter_width = 0.12,
  points = TRUE,
  borders = TRUE,
  legends = FALSE,
  seed = NULL,
  forget = FALSE
)
```

Arguments

...	A generic aesthetic parameter across points and the boxplot. This is optional.
bar.params	List of parameters for the bar graph, such as color, alpha, fill etc
err.params	List of parameters for the error bar, such as color, size, alpha etc
point.params	List of parameters for individual points, such as color, alpha, fill etc
errorbar_type	This argument determines the errorbar type. If it is set to 'se', standard error bar will be shown. If it is set to 'sd' (default), the error bar will display standard deviation. If it is set to 'ci', the error bar will display 95% confidence interval.
point_jitter_width	A numerical value that determines the degree of the jitter for each point. If its 0, all the points will have no jitter (aligned along the y-axis).
points	TRUE if points need to be shown. FALSE if points need to be hidden.
borders	If the border needs to be displayed, the input should be TRUE. If the border is not needed, the input should be FALSE.
legends	If the legend needs to be displayed, the input should be TRUE. If the legend is not needed, the input should be FALSE.
seed	Random seed. Requires a number.

forget	Forget the defaults when list() is called for a specific parameter (ex. point.params). Set to TRUE when when users want to map aesthetics to different groups more flexibly.. Set to FALSE by default.

Value

A bar graph generated using ggplot2

Examples

```
library(smplot2)
library(ggplot2)
set.seed(1) # generate random data
day1 = rnorm(16,2,1)
day2 = rnorm(16,5,1)
Subject <- rep(paste0('S',seq(1:16)), 2)
Data <- data.frame(Value = matrix(c(day1,day2),ncol=1))
Day <- rep(c('Day 1', 'Day 2'), each = length(day1))
df <- cbind(Subject, Data, Day)

# with aesthetic defaults of smplot
ggplot(data = df, mapping = aes(x = Day, y = Value, color = Day)) +
sm_bar() +
scale_color_manual(values = sm_color('blue','orange'))
```

---

sm_bland_altman *A Bland Altman plot*

Description

This function generates a Bland-Altman plot. This function requires two paired data sets as input (same length), and uses sm_statBlandAlt() to compute statistical values necessary for a Bland Altman plot. For more information on these values, please type ?sm_statBlandAlt.

The plot automatically uses sm_classic() theme. The upper dashed line indicates the upper limit (mean_diff + 1.96*sd), the middle dashed line indicates the mean difference between the two samples, and the lower dashed line indicates the lower limit (mean_diff - 1.96*sd).

To add a legend, you will need to add sm_classic(legends = TRUE). To customise the figure, you can add more geom objects.

Usage

```
sm_bland_altman(first, second, point_size = 3.3, diff_ci = TRUE, ...)
```

Arguments

first	Data from the first repetition/session
second	Data from the second repetition/session
point_size	The size of the individual points. The default is set to 3.3.
diff_ci	If set TRUE, then it will draw a shaded region that represents the 95 confidence interval of the difference between the two sessions from one-sample t-test. If the region (i.e. confidence interval) overlaps with zero, then there is no significant bias/difference between the two sessions/datasets. If it does not overlap with 0, then the measurement variability is significantly large.
...	Parameters of geom_point(), such as 'color', 'fill', 'shape', etc.

Value

Prints a figure, which is the Bland-Altman plot (ggplot2 object).

Examples

```
library(smplot2)
library(tibble)

first <- rnorm(20)
second <- rnorm(20)
df <- as_tibble(cbind(first,second))
sm_bland_altman(df$first, df$second)
# when all 3 dashed lines are not shown, extend the range of the y-axis.
```

---

sm_boxplot *A boxplot with jittered individual points*

Description

A boxplot with jittered individual points

Usage

```
sm_boxplot(
  ...,
 boxplot.params = list(notch = FALSE, fill = "gray95", color = "black", size = 0.5,
    width = 0.5, outlier.shape = NA),
  point.params = list(alpha = 0.65, size = 2),
  point_jitter_width = 0.12,
  points = TRUE,
  borders = TRUE,
  legends = FALSE,
  seed = NULL,
  forget = FALSE
)
```

Arguments

...	A generic aesthetic parameter across points and the boxplot. This is optional.
boxplot.params	List of parameters for boxplot, such as color, alpha, fill etc
point.params	List of parameters for individual points, such as color, alpha, fill etc
point_jitter_width	A numerical value that determines the degree of the jitter for each point. If its 0, all the points will have no jitter (aligned along the y-axis).
points	TRUE if points need to be shown. FALSE if points need to be hidden.
borders	If the border needs to be displayed, the input should be TRUE. If the border is not needed, the input should be FALSE.
legends	If the legend needs to be displayed, the input should be TRUE. If the legend is not needed, the input should be FALSE.
seed	Random seed
forget	Forget the defaults when list() is called for a specific parameter (ex. point.params). Set to TRUE when when users want to map aesthetics to different groups more flexibly.. Set to FALSE by default.

Value

A boxplot generated using ggplot2

Examples

```
library(ggplot2)
library(smplot2)
set.seed(1) # generate random data
day1 = rnorm(16,2,1)
day2 = rnorm(16,5,1)
Subject <- rep(paste0('S',seq(1:16)), 2)
Data <- data.frame(Value = matrix(c(day1,day2),ncol=1))
Day <- rep(c('Day 1', 'Day 2'), each = length(day1))
df <- cbind(Subject, Data, Day)

# with the default aesthetics of smplot
ggplot(data = df, mapping = aes(x = Day, y = Value, color = Day)) +
sm_boxplot() +
  scale_color_manual(values = sm_color('blue','orange'))

# Without the default aesthetics of smplot

ggplot(data = df, mapping = aes(x = Day, y = Value, color = Day)) +
sm_boxplot(boxplot.params = list()) +
scale_color_manual(values = sm_color('blue','orange'))
```

sm_ci *Confidence interval*

Description

This function computes the confidence interval.

Usage

```
sm_ci(data, alpha = 0.05, low = TRUE)
```

Arguments

data	Numerical vector of data
alpha	Default is set to 0.05, so that 95% confidence interval is computed.
low	If its TRUE, it will compute the low tail of the confidence interval. If its FALSE, it will compute the high tail of the confidence interval.

Value

Prints a double vector that is a single end of the specified confidence interval.

Examples

```
library(smplot2)
set.seed(1)

a <- rnorm(100,1,1)
sm_ci(a)
sm_ci(a, low=FALSE)
```

sm_classic *A SM classical theme.*

Description

It has x and y axis but no grids.

Usage

```
sm_classic(legends = FALSE)
```

Arguments

legends	If the legend needs to be displayed, the input should be TRUE. If the legend is not needed, the input should be FALSE.

Value

Returns a background theme as a ggplot2 object.

Examples

```
library(ggplot2)
library(smplot2)
ggplot(data = mpg) +
geom_point(mapping = aes(x = displ, y = hwy, color = class)) +
sm_classic()
```

`sm_color` *SM custom palette of colors*

Description

This is a custom color palette that SM recommends for data visualization. It returns up to 20 different colors with a high visibility.

Usage

```
sm_color(...)
```

Arguments

`...`	The input has to be a character string.of a color name. There are 20 colors available from the SM palette: ''blue'',''crimson'', ''green'', ''purple'', ''orange'', ''skyblue'', ''pink'', ''limegreen'', ''lightpurple'', ''brown'', ''red'', ''lightbrown'', ''asparagus'', ''viridian'', ''darkred'', ''lightblue'', ''light blue'', ''wine'', ''yellow'', ''lightgreen''

Value

A character/string of hex codes

Examples

```
library(smplot2)
sm_color('crimson')

sm_color('crimson', 'green', 'blue')
```

sm_common_axis *A function to plot panels with common x- and y- axes*

Description

This function is used to create a composite figure.

Usage

```
sm_common_axis(location, hmargin = 1, wmargin = 1)
```

Arguments

location	Location of the panel. ''topleft'': removes x-axis title, x-axis ticklabel, y-axis title. ''topright'': removes x-axis title, x-axis ticklabel, y-axis title, y-axis ticklabel. ''bottomleft'': removes x-axis title, y-axis title. ''bottomright'': removes x-axis title, y-axis title, y-axis ticklabel. ''topcenter'': removes x-axis title, x-axis ticklabel, y-axis title, y-axis ticklabel. ''bottomcenter'': removes x-axis title, y-axis title, y-axis ticklabel. ''single'': keeps all ticks but removes title ''centerleft'' : removes some ticks and titles ''centerright'' : removes some ticks and titles ''center'': removes everything
hmargin	The amount of height of blank space between subplots. It sets the size of the empty space (i.e., margin) between panels. T he default is set to 1, which should reduce the empty space (right and left side of each panel) between the panels.
wmargin	The amount of width of blank space between subplots. It sets the size of the empty space (i.e., margin) between panels. T he default is set to 1, which should reduce the empty space (right and left side of each panel) between the panels.

Value

Returns a ggplot2 output with ticks removed.

Examples

```
library(ggplot2)
library(smplot2)
set.seed(1) # generate random data
day1 = rnorm(16,2,1)
day2 = rnorm(16,5,1)
Subject <- rep(paste0('S',seq(1:16)), 2)
Data <- data.frame(Value = matrix(c(day1,day2),ncol=1))
Day <- rep(c('Day 1', 'Day 2'), each = length(day1))
df <- cbind(Subject, Data, Day)

# with aesthetic defaults of smplot
ggplot(data = df, mapping = aes(x = Day, y = Value, color = Day)) +
sm_bar() +
scale_color_manual(values = sm_color('blue','orange')) +
```

```
sm_common_axis('bottomleft')
```

`sm_common_legend` *Creating a common legend for subplots on a separate panel*

Description

Creating a common legend for subplots on a separate panel

Usage

```
sm_common_legend(
  x = 0.5,
  y = 0.5,
  title = FALSE,
  direction = "vertical",
  border = TRUE,
  legend_spacing = 0.5,
  border_color = "black",
  textRatio = 1
)
```

Arguments

`x`	Location of the legend along the x-axis. Default is the middle origin (0.5).
`y`	Location of the legend along the y-axis. Default is the middle origin (0.5).
`title`	Title of the legend. Input should be string
`direction`	Direction of the legend: 'horizontal' or 'vertical'.
`border`	If set TRUE, border around the legend will be created. If set FALSE, the border will be removed.
`legend_spacing`	Spacing within the legend.
`border_color`	Color of the legend border
`textRatio`	Size of the text relative to the plot's default. It has been set to 1.2. The larger the textRatio, the larger the texts in the legend.

Value

It prints a legend on a blank plot. It can be used to create a common legend for subplots.

Examples

```
library(ggplot2)
library(smplot2)
ggplot(data = mtcars, mapping = aes(x = drat, y = mpg,
fill = as.factor(cyl))) +
 geom_point(shape = 21, color = 'white', size = 3) +
 sm_common_legend(x = .5, y = 0.5 , direction='horizontal',
                  border=FALSE)
```

`sm_common_title` *Common title for combined subplots*

Description

Common title for combined subplots

Usage

```
sm_common_title(title = "", size = 17, x = 0.5, y = 0.5, fontface = "bold")
```

Arguments

`title`	The input should be string.
`size`	Text size of the title.
`x`	Location of the title along the x-axis. Default is the middle origin (0.5).
`y`	Location of the legend along the y-axis. Default is the middle origin (0.5).
`fontface`	The default is to set the text of the title as bold. This can be changed, to either "plain", "bold", "italic", "bold.italic" .

Value

It prints title on a blank layer of ploting.

Examples

```
library(smplot2)
sm_common_title('My title')
```

sm_common_xlabel *Common x-axis label (title) for combined subplots*

Description

Common x-axis label (title) for combined subplots

Usage

```
sm_common_xlabel(label = "", size = 17, x = 0.5, y = 0.5, fontface = "plain")
```

Arguments

label	The input should be string.
size	Text size of the label.
x	Location of the label along the x-axis. Default is the middle origin (0.5).
y	Location of the label along the y-axis. Default is the middle origin (0.5).
fontface	The default is to set the text of the title as plain This can be changed, to either "plain", "bold", "italic", "bold.italic" .

Value

It returns a layer with the specified common x-axis label for combined plot.

Examples

```
library(smplot2)
sm_common_xlabel('My x-axis')
```

sm_common_ylabel *Common y-axis label (title) for combined subplots*

Description

Common y-axis label (title) for combined subplots

Usage

```
sm_common_ylabel(
  label = "",
  size = 17,
  x = 0.5,
  y = 0.52,
  fontface = "plain",
  angle = 90
)
```

Arguments

label	The input should be string.
size	Text size of the label.
x	Location of the label along the x-axis. Default is the middle origin (0.5).
y	Location of the label along the y-axis. Default is the middle origin (0.5).
fontface	The default is to set the text of the title as plain This can be changed, to either "plain", "bold", "italic", "bold.italic" .
angle	Orientation of the y-axis title. Default is 90 degrees.

Value

It returns a layer with the specified common y-axis label for combined plot.

Examples

```
library(smplot2)
sm_common_ylabel('My y-axis')
```

---

sm_corr_avgErr *Superimposition of the average point with horizontal and vertical error bars in the correlation plot*

Description

Superimposition of the average point with horizontal and vertical error bars in the correlation plot

Usage

```
sm_corr_avgErr(
  data,
  x,
  y,
  point.params = list(size = 2.5),
  errh.params = list(height = 0),
  errv.params = list(width = 0),
  errorbar_type = "se",
  ...
)
```

Arguments

data	Data frame variable that is used for plotting.
x	Column of the data frame that represents the x-axis.
y	Column of the data frame that represents the y-axis.
point.params	List of parameters for the mean point, such as color, alpha, fill etc

errh.params	List of parameters for the horizontal error bar, such as color, alpha, fill etc
errv.params	List of parameters for the vertical points, such as color, alpha, fill etc
errorbar_type	This argument determines the error bar type. If it is set to 'se' , standard error bar will be shown. If it is set to 'sd', the error bar will display standard deviation. If it is set to 'ci' (default), the error bar will display 95% confidence interval.
...	A generic aesthetic parameter across points and error bars. This is optional.

Value

A point with error bars representing the average will be returned.

Examples

```
library(smplot2)
library(ggplot2)
ggplot(data = mtcars, mapping = aes(x = drat, y = mpg)) +
geom_point(shape = 21, size = 3) +
 sm_corr_avgErr(mtcars, drat,mpg, errorbar_type = 'se',
                color = sm_color('red'))
```

---

sm_effsize *Cohen's d - effect size*

Description

Cohen's d is a measure of the effect size. It is often reported with p-values (ex. from a t-test or posthoc pairwise comparisons).

Usage

```
sm_effsize(group1, group2, absolute = TRUE)
```

Arguments

group1	Numeric vector containing data from one sample (i.e., group 1) that is to be compared with another group.
group2	Numeric vector containing data from another sample (i.e., group 2) that is to be compared with the former group.
absolute	If set TRUE, the function will print the absolute value of the effect size. If set FALSE, the function will print effect size of group2 - group1. For example, it will be positive if group2 has a larger mean than group 1.

Value

Returns a double vector that is the effect size between two samples.

Examples

```
library(smplot2)
group1 <- rnorm(10,0,1)
group2 <- rnorm(10,1,1)
sm_effsize(group1, group2)
```

`sm_forest` *Forest plot*

Description

Forest plot

Usage

```
sm_forest(
  ...,
  point.params = list(size = 2.5, alpha = 0.3),
  avgPoint.params = list(size = 5.5, shape = 18),
  err.params = list(color = "black"),
  ref.params = list(size = 0.4, color = "gray80", linetype = "dashed"),
  xintercept = 0,
  sep_level = 2,
  point_jitter_width = 0,
  errorbar_type = "ci",
  points = TRUE,
  refLine = TRUE,
  borders = TRUE,
  legends = FALSE,
  seed = NULL,
  forget = FALSE
)
```

Arguments

`...`	A generic aesthetic parameter across points, lines and error bars. This is optional.
`point.params`	List of parameters for individual points, such as color, alpha, fill etc
`avgPoint.params`	List of parameters for the average point, such as color, alpha, fill etc
`err.params`	List of parameters for the error bar from the average point, such as color, alpha etc
`ref.params`	List of parameters for the vertical reference line, such as color, alpha etc
`xintercept`	Location of the vertical reference line along the x coordinate.

sep_level	A numerical value that controls the level of the separation between the individual points and the average point. If it's 0, all of these are clustered together. If it's higher (and more positive), the text annotations will increasingly go below the mean point. Default is set to 2. The values can be negative so that the points can be above the mean point. There is no limit of the range for this argument.
point_jitter_width	A numerical value that determines the degree of the jitter for each point. If its 0, all the points will have no jitter (aligned along the y-axis).
errorbar_type	This argument determines the error bar type. If it is set to 'se' , standard error bar will be shown. If it is set to 'sd', the error bar will display standard deviation. If it is set to 'ci' (default), the error bar will display 95% confidence interval.
points	If points is set TRUE, individual points are shown. If FALSE, they are not shown.
refLine	If it is set TRUE, the reference line at a specified location along the x-axis is shown. If it is set FALSE, it is not shown.
borders	If the border needs to be displayed, the input should be TRUE. If the border is not needed, the input should be FALSE.
legends	If the legend needs to be displayed, the input should be TRUE. If the legend is not needed, the input should be FALSE.
seed	Random seed
forget	Forget the defaults when list() is called for a specific parameter (ex. point.params). Set to TRUE when when users want to map aesthetics to different groups more flexibly.. Set to FALSE by default.

Value

A forest plot generated using ggplot2

Examples

```
library(smplot2)
library(ggplot2)

day1 = rnorm(20,0,1)
day2 = rnorm(20,5,1)
day3 = rnorm(20,6,1.5)
day4 = rnorm(20,7,2)
Subject <- rep(paste0('S',seq(1:20)), 4)
Data <- data.frame(Value = matrix(c(day1,day2,day3,day4),ncol=1))
Day <- rep(c('Day 1', 'Day 2', 'Day 3', 'Day 4'), each = length(day1))
df2 <- cbind(Subject, Data, Day)

ggplot(data = df2, aes(x = Value, y = Day, color = Day, fill = Day)) +
 sm_forest(sep_level = 2, point_jitter_width = .12,
           errorbar_type = 'ci',
           point.params = list(alpha=0.2)) +
  scale_color_manual(values = sm_palette(4))
```

`sm_forest_annot` *Annotation of the error range on the forest plot*

Description

Annotation of the error range on the forest plot

Usage

```
sm_forest_annot(
  data,
  x,
  y,
  errorbar_type = "ci",
  text.params = list(size = 4, color = "black"),
  sep_level = 2,
  ...
)
```

Arguments

`data`	Data frame variable that is used for plotting.
`x`	Column of the data frame that represents the x-axis.
`y`	Column of the data frame that represents the y-axis.
`errorbar_type`	This argument determines the errorbar type. If it is set to 'se', standard error bar will be shown. If it is set to 'sd' (default), the error bar will display standard deviation. If it is set to 'ci', the error bar will display 95% confidence interval.
`text.params`	List of parameters for the text annotation, such as color, size etc
`sep_level`	A numerical value that controls the level of the separation between the text annotation and the average point. If it's 0, all of these are clustered together. If it's higher (and more positive), the text annotations will increasingly go above the mean point. Default is set to 2. The values can be negative so that the texts can be below the mean point. There is no limit of the range for this argument. Ideally, this should equal to the sep_level in sm_forest().
`...`	Parameters for the text annotation, such as size and color etc.

Value

Annotations showing the range of uncertainty will printed on the forest plot.

Examples

```
library(ggplot2)
library(smplot2)

day1 = rnorm(20,0,1)
```

```
day2 = rnorm(20,5,1)
day3 = rnorm(20,6,1.5)
day4 = rnorm(20,7,2)
Subject <- rep(paste0('S',seq(1:20)), 4)
Data <- data.frame(Value = matrix(c(day1,day2,day3,day4),ncol=1))
Day <- rep(c('Day 1', 'Day 2', 'Day 3', 'Day 4'), each = length(day1))
df2 <- cbind(Subject, Data, Day)

ggplot(data = df2, aes(x = Value, y = Day, color = Day)) +
 sm_forest(point_jitter_width = 0.12, sep_level = 3) +
 scale_color_manual(values = sm_palette(4)) +
 sm_forest_annot(data = df2, x = Value, y = Day, sep_level = 3)
```

---

sm_hgrid *Minimalistic theme of horizontal major grids*

Description

A graph with a horizontal grid is plotted. Border can be added or removed. This is useful for plotting a bar graph.

Usage

```
sm_hgrid(legends = FALSE, borders = TRUE)
```

Arguments

legends	If the legend needs to be displayed, the input should be TRUE. If the legend is not needed, the input should be FALSE.
borders	If the border needs to be displayed, the input should be TRUE. If the border is not needed, the input should be FALSE.

Value

Returns a background theme with major horizontal grids (ggplot2 output).

Examples

```
library(ggplot2)
library(smplot2)
ggplot(data = mpg) +
geom_point(mapping = aes(x = displ, y = hwy, color = class)) +
sm_hgrid()
```

sm_hist *Histogram with kernel density estimation (Gaussian) and rugs*

Description

Histogram with kernel density estimation (Gaussian) and rugs

Usage

```
sm_hist(
  ...,
 hist.params = list(binwidth = 1/2, fill = sm_color("blue"), color = "white", alpha =
    0.4),
 density.params = list(color = sm_color("blue"), size = 0.8, fill = "transparent"),
  rug.params = list(color = sm_color("blue"), alpha = 0.8, size = 0.4),
  histogram = TRUE,
  density = TRUE,
  rug = TRUE,
  borders = FALSE,
  legends = FALSE,
  forget = FALSE
)
```

Arguments

...	A generic aesthetic parameter across points and the boxplot. This is optional.
hist.params	List of parameters for the histogram, such as binwidth, color, alpha, fill etc.
density.params	List of parameters for the density estimation, such as color, size, alpha etc
rug.params	List of parameters for the rugs, such as color, size, alpha etc
histogram	TRUE if the histogram needs to be shown. FALSE if the histogram needs to be hidden.
density	TRUE if the density plot needs to be shown. FALSE if the density plot needs to be hidden.
rug	TRUE if the rugs need to be shown. FALSE if the rugs need to be hidden.
borders	If the border needs to be displayed, the input should be TRUE. If the border is not needed, the input should be FALSE.
legends	If the legend needs to be displayed, the input should be TRUE. If the legend is not needed, the input should be FALSE.
forget	Forget the defaults when list() is called for a specific parameter. Set to TRUE when when users want to map aesthetics to different groups more flexibly.. Set to FALSE by default.

Value

Returns a histogram generated using ggplot2.

Examples

```
library(ggplot2)
library(smplot2)
set.seed(2)
data=data.frame(value=rnorm(1000))
data2 = data.frame(value=rnorm(1000,5,1))

data$day <- 'day1'
data2$day <- 'day2'
rbind(data,data2) -> df

ggplot(data = data, aes(x=value)) +
sm_hist()

ggplot(data = df, aes(x=value, fill=day, color = day)) +
sm_hist(hist.params = list(binwidth = 1/2, alpha = 0.3),
       density.params = list(fill='transparent', size = 0.8),
       rug.params = list(alpha = 0.8)) +
 scale_color_manual(values = sm_palette(2)) +
  scale_fill_manual(values = sm_palette(2))
```

`sm_hvgrid` *Minimalistic theme with major horizontal and vertical grids*

Description

This theme has major vertical and horizontal grids. This is useful for plotting correlations. sm_corr_theme() is exactly the same as sm_hvgrid().

Usage

```
sm_hvgrid(legends = TRUE, borders = TRUE)
```

Arguments

`legends`	If the legend needs to be displayed, the input should be TRUE. If the legend is not needed, the input should be FALSE.
`borders`	If the border needs to be displayed, the input should be 'TRUE'. If the border is not needed, the input should be 'FALSE'.

Value

Returns a background theme that has both horizontal and vertical major grids (ggplot2 output).

Examples

```
library(ggplot2)
library(smplot2)
ggplot(data = mpg) +
geom_point(mapping = aes(x = displ, y = hwy, color = class)) +
sm_hvgrid()
```

`sm_hvgrid_minor` *A theme with horizontal and vertical major and minor grids*

Description

This theme has vertical and horizontal grids.

Usage

```
sm_hvgrid_minor(legends = TRUE, borders = TRUE)
```

Arguments

`legends`	If the legend needs to be displayed, the input should be TRUE. If the legend is not needed, the input should be FALSE.
`borders`	If the border needs to be displayed, the input should be 'TRUE'. If the border is not needed, the input should be 'FALSE'.

Value

Returns a background theme that has both horizontal and vertical major and minor grids (ggplot2 output).

Examples

```
library(ggplot2)
library(smplot2)
ggplot(data = mpg) +
geom_point(mapping = aes(x = displ, y = hwy, color = class)) +
sm_hvgrid_minor()
```

sm_minimal *SM minimal theme (no grid) with borders.*

Description

This theme has no major grid.

Usage

```
sm_minimal(legends = FALSE)
```

Arguments

legends	If the legend needs to be displayed, the input should be TRUE. If the legend is not needed, the input should be FALSE.

Value

Returns a background theme that has no grids (ggplot2 output).

Examples

```
library(ggplot2)
library(smplot2)
ggplot(data = mpg) +
geom_point(mapping = aes(x = displ, y = hwy, color = class)) +
sm_minimal()
```

sm_palette *SM custom palette of colors*

Description

This is a custom color palette that SM recommends for data visualization. It returns up to 20 different colors with a high contrast.

Usage

```
sm_palette(colorNum = 10)
```

Arguments

colorNum	Number of colors (1-20).

Value

Returns a hex code in string vector. The input determines the length of the output.

Examples

```
library(smplot2)
#' sm_palette(3) # returns 3 colors
```

`sm_panel_label` *Writing a label for each panel of a combined figure*

Description

Writing a label for each panel of a combined figure

Usage

```
sm_panel_label(
  all_plots,
  x,
  y,
  panel_tag = "1",
  panel_pretag,
  panel_posttag,
  text_size = 5.5,
  text_color = "black",
  fontface = "plain",
  ...
)
```

Arguments

`all_plots`	all_plots should be a list vector, which should contain all panels that are to be combined into one figure.
`x`	Location of the label along the x-axis (0 to 1). 0.5 is the middle origin.
`y`	Location of the label along the y-axis (0 to 1). 0.5 is the middle origin.
`panel_tag`	A character vector that defines how each panel is enumerated. Options include: 'a', 'A', '1', 'I' or 'i'.'a' is for lowercase letters. 'A' is for uppercase letters. '1' is for integers. 'I' is for upper case roman numerals. 'i' is for lower case roman numerals. Each panel will display a unique string based on the set enumeration.
`panel_pretag`	A character vector that is identical across panels BEFORE the panel_tag.
`panel_posttag`	A character vector that is identical across panels AFTER the panel_tag.
`text_size`	Text size of the panel label
`text_color`	Text color of the panel label

fontface	Fontface of the panel label. Options include "plain", "bold", "italic" and others that are provided by ggplot2.
...	Additional parameters for adjusting the appearance of the sticker. Same parameters for annotate() from ggplot2.

Value

It returns a list of plots with panel labels.

Examples

```
library(ggplot2)
library(smplot2)

ggplot(data = mtcars, mapping = aes(x = drat, y = mpg)) +
geom_point(shape = 21, fill = '#0f993d', color = 'white',
          size = 3) -> p1

ggplot(data = mtcars, mapping = aes(x = drat, y = mpg)) +
 geom_point(shape = 21, fill = '#0f993d', color = 'white', size = 3) +
 sm_hvgrid() -> p2

sm_panel_label(list(p1,p2), x = 0.1, y = 0.9,
              panel_tag ='1', panel_pretag = 'S', text_size = 4, text_color = 'black')
```

---

sm_plot_clean *Remove xticklabels and yticklabels in selected panels for proper subplotting*

Description

Remove xticklabels and yticklabels in selected panels for proper subplotting

Usage

```
sm_plot_clean(all_plots, ncol, nrow, wmargin = wmargin, hmargin = hmargin)
```

Arguments

all_plots	all_plots should be list, which should contain all panels that are to be combined into one figure.
ncol	Number of columns in the combined plot
nrow	Number of rows in the combined plot
wmargin	The amount of width of blank space between subplots. It sets the size of the empty space (i.e., margin) between panels. T he default is set to 1, which should reduce the empty space (right and left side of each panel) between the panels.

hmargin The amount of height of blank space between subplots. It sets the size of the empty space (i.e., margin) between panels. T he default is set to 1, which should reduce the empty space (right and left side of each panel) between the panels.

Value

Returns a list of plots with new layouts.

Examples

```
library(smplot2)
library(ggplot2)
ggplot(data = mtcars, mapping = aes(x = drat, y = mpg)) +
  geom_point(shape = 21, fill = '#0f993d', color = 'white',
             size = 3) -> p1

ggplot(data = mtcars, mapping = aes(x = drat, y = mpg)) +
  geom_point(shape = 21, fill = '#0f993d', color = 'white', size = 3) +
  sm_hvgrid() -> p2

sm_plot_clean(list(p1,p2), ncol=2,nrow=1,wmargin=-2, hmargin=-2)
```

---

sm_pointplot *Point plot with optional shadow*

Description

This is a common plot with mean point, standard error (se, sd or 95 uniquely shadow, which is a faint display of individual points behind the mean.

Usage

```
sm_pointplot(
  ...,
  avgPoint.params = list(size = 2.5),
  avgLine.params = list(linewidth = 1),
  point.params = list(alpha = 0.35, color = "gray", fill = "gray"),
  line.params = list(alpha = 0.35, color = "gray"),
  err.params = list(linewidth = 1),
  errorbar_type = "se",
  show_shadow = FALSE,
  group = NULL,
  borders = TRUE,
  legends = FALSE,
  forget = FALSE
)
```

Arguments

...	A generic aesthetic parameter across points, lines and errorbars. This is optional. This will be extremely useful for dodging each line using position_dodge().
avgPoint.params	List of parameters for the average point, such as color, alpha, fill etc
avgLine.params	List of parameters for the average line, such as color, alpha etc
point.params	List of parameters for the points in the shadow, such as color, alpha, fill etc
line.params	List of parameters for the lines in the shadow, such as color, alpha etc
err.params	List of parameters for the error bar from the average plot, such as color, alpha etc
errorbar_type	This argument determines the errorbar type. If it is set to 'se', standard error bar will be shown. If it is set to 'sd' (default), the error bar will display standard deviation. If it is set to 'ci', the error bar will display 95% confidence interval.
show_shadow	If it is TRUE, it will show the shadow. If it is FALSE, it will not show the shadow (default).
group	If show_shadow = TRUE, this argument is required. This is the variable that each plot from the shadow should be grouped along aesthetically. It should be grouped for each individual observation, ex. sm_pointplot(group = Subject), whereby Subject is the column that holds identifers for each observation.
borders	If the border needs to be displayed, the input should be TRUE. If the border is not needed, the input should be FALSE.
legends	If the legend needs to be displayed, the input should be TRUE. If the legend is not needed, the input should be FALSE.
forget	Forget the defaults when list() is called for a specific parameter (ex. point.params). Set to TRUE when when users want to map aesthetics to different groups more flexibly.. Set to FALSE by default.

Value

Returns a pointplot generated using ggplot2.

Examples

```
library(smplot2)
library(ggplot2)
ggplot(data = mtcars, mapping = aes(x = cyl, y = mpg)) +
sm_pointplot()
```

sm_power *Post-hoc power analysis using two-sample or paired t-test*

Description

Post-hoc power analysis using two-sample or paired t-test

Usage

```
sm_power(group1, group2, paired, sig.level = 0.05, power = 0.8)
```

Arguments

group1	Numeric vector containing data from one sample (i.e., group 1) that is to be compared with another group.
group2	Numeric vector containing data from another sample (i.e., group 2) that is to be compared with the former group.
paired	A logical indicating whether your two samples (group1 and group2) are paired.
sig.level	Significance level (Type I error probability). Default is set to 0.05.
power	Power of test (1 minus Type II error probability). Default is set to 0.8.

Value

Returns a result with a class of "power.htest" from the pwr package.

Examples

```
library(smplot2)
group1 <- rnorm(10,0,1)
group2 <- rnorm(10,1,1)
sm_power(group1, group2, paired = TRUE)
```

sm_put_together *Combining figures together*

Description

This function works best if all_plots argument (the list input) contains plots that have tick labels on both x and y axes; this information will be used to optimize the layout of the composite figure.

Usage

```
sm_put_together(
  all_plots,
  title,
  xlabel,
  ylabel,
  legend,
  ncol,
  nrow,
  xlabel2,
  ylabel2,
  tickRatio,
  panel_scale = 0.9,
  wRatio,
  hRatio,
  hmargin = 0,
  wmargin = 0,
  remove_ticks = "some",
  wRatio2,
  hRatio2,
  labelRatio = 1
)
```

Arguments

all_plots	all_plots should be list, which should contain all panels that are to be combined into one figure.
title	Title layer that will determine the main title of the combined plot. This is created using sm_common_title(). Optional argument. Users can also supply character string here instead.
xlabel	xlabel layer that will determine the label of the combined plot's x-axis. This is created using sm_common_xlabel(). Optional argument. Users can also supply character string here instead.
ylabel	ylabel layer that will determine the label of the combined plot's y-axis. This is created using sm_common_ylabel(). Optional argument. Users can also supply character string here instead.
legend	ggplot() layer that has legend. Optional argument.
ncol	Number of columns in the combined plot
nrow	Number of rows in the combined plot
xlabel2	2nd xlabel layer that will determine the label of the combined plot's secondary x-axis. This is created using sm_common_xlabel(). Optional argument. Users can also supply character string here instead.
ylabel2	2nd ylabel layer that will determine the label of the combined plot's y-axis. This is created using sm_common_ylabel(). Optional argument. Users can also supply character string here instead.

`tickRatio` Relative size of the ticks to the default aesthetics of the thematic functions (ex. sm_hgrid()). By default, it adjusts tickRatio based on the given plot. But this can be overwritten if the input is supplied (ex. try 1.4 to begin with). FOr example, 1.4x means that it is 1.4x larger than the tick size of a single given plot.

`panel_scale` Scale of the panel. Default is set to 0.9 to reduce empty space within and around each panel. The user can set to a value from 0 to 1 to see what happens to the spacing within each panel and between panels.

`wRatio` This adjusts the ratio of the width of the first column to those of other columns. By default, it adjusts wRatio based on the given plot. However, this can be overwritten if the input is supplied. If the value is larger than 1, then it will be wider than that of other columns. Users are encouraged to adjust this value because different computers can show different looking outputs.

`hRatio` This adjusts the ratio of the height of the last row to those of other rows By default, it adjusts hRatio based on the given plot. However, this can be overwritten if the input is supplied. If the value is larger than 1, then it will be taller than that of other columns. Users are encouraged to adjust this value because different computers can show different looking outputs.

`hmargin` The amount of height of blank space between subplots. It sets the size of the empty space (i.e., margin) between panels. T he default is set to 0. If its positive, the blank spacing will increase. If its negative, it will get reduced between panels.

`wmargin` The amount of width of blank space between subplots. It sets the size of the empty space (i.e., margin) between panels. T he default is set to 0. If its positive, the blank spacing will increase. If its negative, it will get reduced between panels.

`remove_ticks` If set to 'some', x-axis ticks and y-axis ticks will be removed in inner plots. If set to 'all', then all panels' ticks will be removed. If set to 'none', then all panels' ticks will be kept.

`wRatio2` This adjusts the ratio of the width of the last column to those of other columns. By default, it adjusts wRatio2 based on the given plot. However, this can be overwritten if the input is supplied. If the value is larger than 1, then it will be wider than that of other columns. Users are encouraged to adjust this value because different computers can show different looking outputs.

`hRatio2` This adjusts the ratio of the height of the first row to those of other rows By default, it adjusts hRatio2 based on the given plot. However, this can be overwritten if the input is supplied. If the value is larger than 1, then it will be taller than that of other columns. Users are encouraged to adjust this value because different computers can show different looking outputs.

`labelRatio` Relative text size of the labels, such as title, xlabel, ylabel, xlabel2 and ylabel2 to its default font size (optimized). This input only changes the size if the inputs are provided as character strings. The default value is 1. If this input is larger than 1.1, then the text size will be larger 1.1x than the default size, which itself is optimized based on the plot's given layout and other information.

Details

The inputs for the axis labels can be created with sm_common_xlabel(), sm_common_ylabel() and sm_common_title(). Alternatively, users can supply character strings directly to sm_put_together() instead. However, this option is not flexible but the function tries its best to find the optimal size and location given the plot information.

Value

Returns a combined figure.

Examples

```
library(smplot2)
library(ggplot2)

ggplot(data = mtcars, mapping = aes(x = drat, y = mpg)) +
geom_point(shape = 21, fill = '#0f993d', color = 'white',
          size = 3) -> p1

ggplot(data = mtcars, mapping = aes(x = drat, y = mpg)) +
  geom_point(shape = 21, fill = '#0f993d', color = 'white', size = 3) +
  sm_hvgrid() -> p2

title <- sm_common_title('My title')
xlabel <- sm_common_xlabel('My x-axis')
ylabel <- sm_common_ylabel('My y-axis')

sm_put_together(list(p1,p2), title=title, xlabel=xlabel,
                ylabel=ylabel, ncol=2,nrow=1)

sm_put_together(list(p1,p2), title='My title', xlabel='My x-axis',
                ylabel='My y-axis', labelRatio = 1.1, ncol=2,nrow=1)
```

---

sm_raincloud *Raincloud plot*

Description

This function visualizes a raincloud plot, which is a combination of jittered points, boxplots and violin plots. The creation of this function has been inspired by the R package called 'raincloudplots' by Jordy van Langen.

This function has been created to allow more customisation than the functions in the raincloudplots package. Also, this function automatically sorts the data given the condition that the x-axis factor levels have been sorted properly.

Usage

```
sm_raincloud(
  ...,
  boxplot.params = list(),
  violin.params = list(alpha = 0.3, color = "transparent"),
  point.params = list(alpha = 1, size = 3, shape = 21, color = "transparent"),
  which_side = "r",
  sep_level = 2,
  point_jitter_width = 0.12,
  vertical = TRUE,
  points = TRUE,
  borders = TRUE,
  legends = FALSE,
  seed = NULL,
  forget = FALSE
)
```

Arguments

`...`	A generic aesthetic parameter across points, boxplot and violin. This is optional.
`boxplot.params`	List of parameters for boxplot, such as color, alpha, fill etc
`violin.params`	List of parameters for violin, such as color, alpha, fill etc
`point.params`	List of parameters for individual points, such as color, alpha, fill etc
`which_side`	String argument to specify the side of the boxplots and violinplots. The options are: 'right' and 'left'. 'mixed' has been removed from smplot due to its lack of usage.
`sep_level`	A numerical value that controls the level of the separation among the boxplot, violin plot and the points. The value can be 0-4. If it's 0, all of these are clustered together. If it's 3, they are all separated. 1 and 2 are somewhere in the middle. Default is set to 2.
`point_jitter_width`	A numerical value that determines the degree of the jitter for each point. If its 0, all the points will have no jitter (aligned along the y-axis).
`vertical`	The orientation of the plots. The default is set to TRUE. If you want the horizontal orientation of the plot, set this argument as FALSE.
`points`	If the points need to be displayed, the input should be TRUE. If the points are not needed, the input should be FALSE.
`borders`	If the border needs to be displayed, the input should be TRUE. If the border is not needed, the input should be FALSE.
`legends`	If the legend needs to be displayed, the input should be TRUE. If the legend is not needed, the input should be FALSE.
`seed`	Random seed
`forget`	Forget the defaults when list() is called for a specific parameter (ex. point.params). Set to TRUE when when users want to map aesthetics to different groups more flexibly.. Set to FALSE by default.

Value

Returns a raincloud plot generated using ggplot2.

Examples

```
library(ggplot2)
library(smplot2)

set.seed(2) # generate random data
day1 = rnorm(20,0,1)
day2 = rnorm(20,5,1)
day3 = rnorm(20,6,1.5)
day4 = rnorm(20,7,2)
Subject <- rep(paste0('S',seq(1:20)), 4)
Data <- data.frame(Value = matrix(c(day1,day2,day3,day4),ncol=1))
Day <- rep(c('Day 1', 'Day 2', 'Day 3', 'Day 4'), each = length(day1))
df2 <- cbind(Subject, Data, Day)

ggplot(data=df2, aes(x = Day, y = Value, color = Day, fill = Day)) +
sm_raincloud() +
 xlab('Day')  +
 scale_fill_manual(values = sm_palette(4))
```

---

sm_slope *A slope chart*

Description

This function generates a slope chart. This is very useful for comparing the effect between two time points.

ggplot()'s mapping has to be quite specific: each observation has to be grouped.

Error bar types can be specified (ci, sd, and se).

Usage

```
sm_slope(
  ...,
  labels,
  group,
  line.params = list(color = "gray53", linewidth = 0.4, alpha = 0.4),
  point.params = list(size = 2.5, shape = 21, color = "white"),
  avgLine.params = list(linewidth = 1),
  avgPoint.params = list(size = 4),
  err.params = list(linewidth = 1),
  xTick.params = list(position = "top", expand = c(0.17, 0.1), drop = FALSE),
```

```
  errorbar_type = "sd",
  many_groups = FALSE,
  show_err = FALSE,
  show_mean = FALSE,
  legends = FALSE,
  forget = FALSE
)
```

Arguments

...	List of parameters for individual points and lines across different elements (except for except for xTick.params), such as color, alpha, fill etc.
labels	Labels for the ticks of the x-axis. This is a required argument. It has to be a single vector containing either one or multiple elements. ex: c('Day 1', 'Day 2')
group	Name of the variable by which the individual data should be grouped
line.params	List of parameters for the individual lines, such as color, alpha etc
point.params	List of parameters for the individual points, such as color, alpha, fill etc
avgLine.params	List of parameters for the average line, such as color, alpha etc
avgPoint.params	List of parameters for the average point, such as color, alpha, fill etc
err.params	List of parameters for the error bar from the average plot, such as color, alpha etc
xTick.params	List of parameters for the x tick from the average plot, such as color, alpha etc
errorbar_type	This argument determines the errorbar type. If it is set to 'se', standard error bar will be shown. If it is set to 'sd' (default), the error bar will display standard deviation. If it is set to 'ci', the error bar will display 95% confidence interval.
many_groups	This argument determines whether the average line can be plotted for each group when multiple groups are plotted at once. If the average line needs to be plotted across all data presented, set this as FALSE. If there are many groups that are presented and that each average line has to be plotted, then set this as TRUE.
show_err	If the error bar needs to be displayed, the input should be TRUE. If the error bar is not needed, the input should be FALSE.
show_mean	If the average plot needs to be displayed, the input should be TRUE. If the average plot is not needed, the input should be FALSE.
legends	If the legend needs to be displayed, the input should be TRUE. If the legend is not needed, the input should be FALSE.
forget	Forget the defaults when list() is called for a specific parameter (ex. point.params). Set to TRUE when when users want to map aesthetics to different groups more flexibly.. Set to FALSE by default.

Value

Returns a slope chart which is a ggplot2 object.

Examples

```
library(ggplot2)
library(smplot2)

set.seed(1) # generate random data
day1 = rnorm(16,2,1)
day2 = rnorm(16,5,1)
Subject <- rep(paste0('S',seq(1:16)), 2)
Data <- data.frame(Value = matrix(c(day1,day2),ncol=1))
Day <- rep(c('Day 1', 'Day 2'), each = length(day1))
df <- cbind(Subject, Data, Day)

ggplot(data=df, aes(x = Day, y = Value, fill = Day)) +
 sm_slope(labels = c('Day 1', 'Day 2'), group = Subject) +
 scale_fill_manual(values=  sm_color('blue','orange'))
ggplot(data = df, aes(x = Day, y = Value, fill = Day)) +
 sm_slope(labels = c('Day 1','Day 2'),group = Subject,
          point.params = list(alpha = 0.3, size = 2.5, color = 'white',
                              shape = 21, fill = sm_color('skyblue')),
          line.params = list(color = sm_color('skyblue'),
                             alpha = 0.3),
          avgPoint.params = list(color='transparent', shape = 21,
                                 size = 4, fill = sm_color('blue')),
          avgLine.params = list(color = sm_color('blue'), linewidth = 1),
          show_mean = TRUE)
```

`sm_slope_all` *Calculating the slope across multiple conditions and subjects*

Description

This function returns a data frame containing slope (from linear regression) from a data frame that contains the original raw data.

The user can use lm() from base R to compute the slope as well.

Usage

```
sm_slope_all(data, subjects, groups, conditions, x, values)
```

Arguments

`data`	Name of the variable that stores the data frame that contains the columns with the specified column names.
`subjects`	The name of the column of the data frame that contains subjects. It must be strings.

groups	The name of the column of the data frame that contains each group. It must be strings.
conditions	The name of the column of the data frame that contains each condition. It must be strings.
x	The name of the column of the data frame that contains the x-axis points/x coordinates from which the slopes can be calculated. It must be strings. The column must not have characters.
values	The name of the column of the data frame that contains the actual data, which are the y-axis points from which the slope can be calculated. It must be strings.

Value

Returns a data frame containing slopes for each subject and each experimental condition and/or group.

Examples

```
library(smplot2)
set.seed(1) # generate random data
day1 = rnorm(16,0,1)
day2 = rnorm(16,5,1)
Subject <- rep(paste0('S',seq(1:16)), 2)
Value <- data.frame(Value = matrix(c(day1,day2),ncol=1))
Day <- rep(c(1,2), each = length(day1))
Condition <- rep('Control', length(day1)*2)
df <- cbind(Subject, Value, Condition, Day)

sm_slope_all(data = df, subjects = 'Subject',values = 'Value',
conditions = 'Condition',x = 'Day')
```

---

sm_slope_mean *A slope chart (with mean) of one group*

Description

This function provides an easy way to plot slope chart with mean. This can also be reproduced using sm_slope().

Usage

```
sm_slope_mean(
  ...,
  labels,
  group,
  main_color = sm_color("blue"),
```

```
  main_shape = 21,
  back_alpha = 0.25,
  line_width = 0.25,
  avgline_width = 1,
  point_size = 2.5,
  avgpoint_size = 4,
  err_width = 1,
  xTick.params = list(position = "top", expand = c(0.17, 0.1), drop = FALSE),
  errorbar_type = "sd",
  show_err = FALSE,
  legends = FALSE
)
```

Arguments

...	List of parameters for individual points and lines across different elements (except for except for xTick.params), such as color, alpha, fill etc.
labels	Labels for the ticks of the x-axis. This is a required argument. It has to be a single vector containing either one or multiple elements. ex: c('Day 1', 'Day 2')
group	Name of the variable by which the individual data should be grouped
main_color	Main color of the slope chart. Shared across points, lines and error bars.
main_shape	Main shape of the points in the slope chart.
back_alpha	Transparency of the shadow (individual lines and points) from the back.
line_width	Line width of the line that connects points in the back shadow
avgline_width	Average's linewidth of the line that connects points in the back shadow
point_size	Size of the points in the back from the shadow
avgpoint_size	Size of the points representing the mean of the data
err_width	Linewidth of the errorbars
xTick.params	List of parameters for the x tick from the average plot, such as color, alpha etc
errorbar_type	This argument determines the errorbar type. If it is set to 'se', standard error bar will be shown. If it is set to 'sd' (default), the error bar will display standard deviation. If it is set to 'ci', the error bar will display 95% confidence interval.
show_err	If the error bar needs to be displayed, the input should be TRUE. If the error bar is not needed, the input should be FALSE.
legends	If the legend needs to be displayed, the input should be TRUE. If the legend is not needed, the input should be FALSE.

Details

This is very useful for comparing the effect between two time points of one group.

ggplot()'s mapping has to be quite specific: each observation has to be grouped.

Error bar types can be specified (ci, sd, and se).

Value

Returns a slope chart which is a ggplot2 object.

Examples

```
library(ggplot2)
library(smplot2)

set.seed(1) # generate random data
day1 = rnorm(16,2,1)
day2 = rnorm(16,5,1)
Subject <- rep(paste0('S',seq(1:16)), 2)
Data <- data.frame(Value = matrix(c(day1,day2),ncol=1))
Day <- rep(c('Day 1', 'Day 2'), each = length(day1))
df <- cbind(Subject, Data, Day)

ggplot(data=df, aes(x = Day, y = Value)) +
 sm_slope_mean(labels = c('Day 1', 'Day 2'), group = Subject, back_alpha = .3,
main_color = sm_color('green'))
```

---

sm_slope_theme *SM plot with a theme appropriate for the slope chart*

Description

In this plot, all aspects except for the left-handed spine are missing. This format is appropriate for the slope chart.

Usage

```
sm_slope_theme(legends = TRUE)
```

Arguments

legends	#' If the legend needs to be displayed, the input should be TRUE. If the legend is not needed, the input should be FALSE.

Value

Returns a background theme that is suitable for a slope chart (ggplot2 output).

Examples

```
library(ggplot2)
library(smplot2)

ggplot(data = mpg) +
geom_point(mapping = aes(x = displ, y = hwy, color = class)) +
sm_slope_theme()
```

`sm_statBlandAlt` *Statistics for a Bland-Altman plot*

Description

Bland-Altman plot is drawn to show measurement variability/reliabiilty of a task. This function requires two paired datasets (same length). It returns a list of difference (by element), mean, standard deviation of the difference, mean difference, upper and lower limits. These values are necessary to draw a Bland Altman plot.

The list returned from this function can be directly used as an argument for sm_bland_altman(), which draws a Bland-Altman plot using ggplot2.

Another output 'data' is a tibble with two columns: 1) Mean across each pair for each element (ex. a mean of the 1st element from the first set and 1st element from the second set), 2) Difference between each pair for every element. The output 'data' should be used as a argument for data in ggplot() when plotting.

Usage

```
sm_statBlandAlt(first, second)
```

Arguments

`first`	Data from the first repetition/session
`second`	Data from the second repetition/session

Value

A list is returned, which has all numerical results that are relevant to drawing a Bland-Altman plot.

Examples

```
library(smplot2)
library(tibble)

first <- rnorm(20)
second <- rnorm(20)
df <- as_tibble(cbind(first,second)) # requires library(tidyverse)
sm_statBlandAlt(df$first, df$second)
```

`sm_statCorr` *Linear regression slope and statistical values (R or R2 and p values) from a paired correlation test.*

Description

This combines two different functions: 1) 'geom_smooth()' from ggplot, and 2) 'stat_cor()' from ggpubr. 'geom_smooth()' is used to fit the best-fit model, whereas 'stat_cor()' is used to print correlation results at an optimized location.

Updates from smplot2 include more flexibility, less input arguments and its pairing with 'sm_hvgrid()' / 'sm_corr_theme()'.

Usage

```
sm_statCorr(
  ...,
  fit.params = list(),
  corr_method = "pearson",
  alternative = "two.sided",
  separate_by = ",",
  label_x = NULL,
  label_y = NULL,
  text_size = 4,
  show_text = TRUE,
  borders = TRUE,
  legends = FALSE,
  r2 = FALSE,
  R2
)
```

Arguments

`...`	Arguments for the properties of regression line, such as 'linetype', 'color', etc. For more information, type ?geom_smooth
`fit.params`	Paramters for the fitted line, such as color, linetype and alpha.
`corr_method`	Method of the correlation test. Options include: 'pearson', 'kendall', or 'spearman'.
`alternative`	Specifies the alternative hypothesis (H1). 'two.sided' is the standard way. 'greater' is a positive association, whereas 'less' is a negative association.
`separate_by`	This marks how the p- and r- values should be separated. The default option is: ',' For more information, check out stat_cor() from the ggpubr package.
`label_x`	Location of the statistical value prints along the figure's x-axis. It asks for a number within the x-axis limit.
`label_y`	Location of the statistical value prints along the figure's y-axis. It requires a number within the y-axis limit.

text_size	Size (numerical value) of the texts from correlation.
show_text	If the statistical result needs to be displayed, the input should be TRUE (default). If the statistical result is not needed, the input should be FALSE.
borders	If the border needs to be displayed, the input should be TRUE. If the border is not needed, the input should be FALSE.
legends	If the legend needs to be displayed, the input should be TRUE. If the legend is not needed, the input should be FALSE.
r2	FALSE or TRUE. TRUE if user wants to compute R2. FALSE if R needs to be computed.
R2	Same as r2.

Value

Plots a best-fitted linear regression on a correlation plot with results from correlation statistical tests.

Examples

```
library(smplot2)
library(ggplot2)
ggplot(data = mtcars, mapping = aes(x = drat, y = mpg)) +
geom_point(shape = 21, fill = '#0f993d', color = 'white', size = 3) +
 sm_statCorr() # computes R

 ggplot(data = mtcars, mapping = aes(x = drat, y = mpg)) +
geom_point(shape = 21, fill = '#0f993d', color = 'white', size = 3) +
 sm_statCorr(R2 = TRUE) # computes R2
```

---

sm_stdErr *Standard error*

Description

Standard error

Usage

```
sm_stdErr(data)
```

Arguments

data	Numerical vector of data.

Value

A double vector is returned with a standard error of the input (given sample).

Examples

```
library(smplot2)
sm_stdErr(rnorm(10,0,1))
```

`sm_vgrid` *Minimalistic theme with vertical major grids*

Description

This theme has major vertical grids.

Usage

```
sm_vgrid(legends = TRUE, borders = TRUE)
```

Arguments

`legends`	If the legend needs to be displayed, the input should be TRUE. If the legend is not needed, the input should be FALSE.
`borders`	If the border needs to be displayed, the input should be TRUE. If the border is not needed, the input should be FALSE.

Value

Returns a background theme with major vertical grids (ggplot2 output).

Examples

```
library(ggplot2)
library(smplot2)

ggplot(data = mpg) +
geom_point(mapping = aes(x = displ, y = hwy, color = class)) +
sm_vgrid()
```

`sm_violin` *A violin plot with jittered individual points*

Description

A violin plot with jittered individual points

Usage

```
sm_violin(
  ...,
  violin.params = list(fill = "gray90", color = "transparent"),
  err.params = list(size = 1.2, linewidth = 1.2),
  point.params = list(alpha = 0.25, size = 2),
  errorbar_type = "sd",
  point_jitter_width = 0.17,
  points = TRUE,
  borders = TRUE,
  legends = FALSE,
  seed = NULL,
  forget = FALSE
)
```

Arguments

`...`	A generic aesthetic parameter across points and the violin plot. This is optional.
`violin.params`	List of parameters for the violin, such as color, alpha, fill etc
`err.params`	List of parameters for the error bar, such as color, size, alpha etc
`point.params`	List of parameters for individual points, such as color, alpha, fill etc
`errorbar_type`	This argument determines the errorbar type. If it is set to 'se', standard error bar will be shown. If it is set to 'sd' (default), the error bar will display standard deviation. If it is set to 'ci', the error bar will display 95% confidence interval.
`point_jitter_width`	A numerical value that determines the degree of the jitter for each point. If its 0, all the points will have no jitter (aligned along the y-axis).
`points`	TRUE if points need to be shown. FALSE if points need to be hidden.
`borders`	If the border needs to be displayed, the input should be TRUE. If the border is not needed, the input should be FALSE.
`legends`	If the legend needs to be displayed, the input should be TRUE. If the legend is not needed, the input should be FALSE.
`seed`	Random seed
`forget`	Forget the defaults when list() is called for a specific parameter (ex. point.params). Set to TRUE when when users want to map aesthetics to different groups more flexibly.. Set to FALSE by default.

Value

Violin plot generated using ggplot2

Examples

```
library(ggplot2)
library(smplot2)
set.seed(1) # generate random data
day1 = rnorm(16,2,1)
day2 = rnorm(16,5,1)
Subject <- rep(paste0('S',seq(1:16)), 2)
Data <- data.frame(Value = matrix(c(day1,day2),ncol=1))
Day <- rep(c('Day 1', 'Day 2'), each = length(day1))
df <- cbind(Subject, Data, Day)
# with aesthetic defaults of smplot
ggplot(data = df, mapping = aes(x = Day, y = Value, color = Day)) +
sm_violin() +
scale_color_manual(values = sm_color('blue','orange'))

# without aesthetic defaults of smplot
ggplot(data = df, mapping = aes(x = Day, y = Value, color = Day)) +
sm_violin(violin.params = list()) +
scale_color_manual(values = sm_color('blue','orange'))
```

Index

sm_add_legend, 2
sm_add_point, 4
sm_add_text, 5
sm_auc, 6
sm_auc_all, 7
sm_bar, 8
sm_bland_altman, 9
sm_boxplot, 10
sm_ci, 12
sm_classic, 12
sm_color, 13
sm_common_axis, 14
sm_common_legend, 15
sm_common_title, 16
sm_common_xlabel, 17
sm_common_ylabel, 17
sm_corr_avgErr, 18
sm_effsize, 19
sm_forest, 20
sm_forest_annot, 22
sm_hgrid, 23
sm_hist, 24
sm_hvgrid, 25
sm_hvgrid_minor, 26
sm_minimal, 27
sm_palette, 27
sm_panel_label, 28
sm_plot_clean, 29
sm_pointplot, 30
sm_power, 32
sm_put_together, 32
sm_raincloud, 35
sm_slope, 37
sm_slope_all, 39
sm_slope_mean, 40
sm_slope_theme, 42
sm_statBlandAlt, 43
sm_statCorr, 44
sm_stdErr, 45
sm_vgrid, 46
sm_violin, 47