Package ‘grafify’

May 30, 2022

Type Package

Title Easy Graphs for Data Visualisation and Linear Models for ANOVA

Version 2.3.0

Description
Easily explore data by generating different kinds of graphs with few lines of code. Use these ggplot() wrappers to quickly draw graphs of scatter/dots with box-whiskers, violins or SD error bars, data distributions, before-after graphs, factorial ANOVA and more. Customise graphs in many ways. Choose from colourblind-friendly 12 discreet , 3 continuous (3 palettes) and 3 divergent colour palettes. Simple code for ANOVA as ordinary (lm()) or mixed-effects linear models (lmer()), including randomised-block or repeated-measures designs. Obtain estimated marginal means and perform post-hoc comparisons on fitted models (via emmeans() wrappers). Also includes small datasets for practicing code and teaching basics before users move on to more complex designs. See vignettes for details on usage <https://grafify-vignettes.netlify.app/>. Citation: <doi:10.5281/zenodo.5136508>.

License GPL (>= 2)

Imports car, emmeans, Hmisc, lme4, lmerTest, magrittr, purrr, stats, tidyr

Depends R (>= 4.0), ggplot2

Encoding UTF-8

Date 2022-05-29

LazyData true

RoxygenNote 7.1.2

Suggests dplyr, knitr, rlang, rmarkdown, pbkrtest, testthat (>= 3.0.0)

URL https://github.com/ashenoy-cmbi/grafify

Config/testthat/edition 3

NeedsCompilation no

Author Avinash R Shenoy [cre, aut] (<https://orcid.org/0000-0001-6228-9303>)

Maintainer Avinash R Shenoy <a.shenoy@imperial.ac.uk>

Repository CRAN

Date/Publication 2022-05-30 09:30:06 UTC

R topics documented:

`colorRampPalette_d` *colorRamPalette_d*

Description

Variant of `colorRampPalette` for sequential use of colours for discrete scales. Thank you linog. Called by other functions in grafify and not generally meant to be called by user.

Usage

```
colorRampPalette_d(colors, ...)
```

Arguments

`colors`	internal
`...`	internal

Value

This is a variant of `colourRampPalette` that generates sequential colours from chosen grafify palettes when called by `graf_col_palette_d`.

colorRamp_d *colourRamp_d*

Description

Variant of `colorRamp` for sequential use of colours for discrete scales. Thank you linog. Called by other functions in grafify and not generally meant to be called by user.

Usage

```
colorRamp_d(
  colors,
  n,
  bias = 1,
  space = c("rgb", "Lab"),
  interpolate = c("linear", "spline"),
  alpha = FALSE
)
```

Arguments

`colors`	internal
`n`	internal
`bias`	internal
`space`	internal
`interpolate`	internal
`alpha`	internal

Value

It generates the required number of sequential colours from chosen grafify palettes when called by `colorRampPalette_d`.

data_1w_death *In vitro experiments measuring percentage cell death in three genotypes of cells.*

Description

These data are from in vitro measurements of death of host cells (measured as percentage of total cells) after infection with three different strains of a pathogenic bacterium, from five independent experiments. The three strains are three levels within the fixed factor Genotype. The five independent experiments are levels within the random variable Experiment. These data can be analysed using linear mixed effects modeling. These data are from Goddard *et al*, Cell Rep, 2019, doi.org/10.1016/j.celrep.2019.03.100

Usage

```
data_1w_death
```

Format

data.frame: 15 obs. of 3 variables.

Experiment Experiment - a random factor with 5 levels "Exp_1","Exp_2"...

Genotype Genotypes - a fixed factor with 3 levels: "WT","KO_1","KO_2".

Death Numerical dependent variable indicating percentage cell death.

`data_2w_Festing` *Data from two-way ANOVA with randomised block design of treatments of strains of mice.*

Description

Data from Festing, ILAR Journal (2014) 55, 472–476 <doi: 10.1093/ilar/ilu045>. These data are suitable for two-way linear mixed effects modelling. The activity of GST (numerical dependent variable) was measured in 4 strains of mice (levels with the fixed factor Strain) either treated or controls (levels within the fixed factor Treatment). Once mouse each was used in two randomised blocks, which is the random factor (Block).

Usage

```
data_2w_Festing
```

Format

data.frame: 16 obs. of 4 variables:

Block A random factor with 2 levels "A" and "B".

Treatment A fixed factor with 2 levels: "Control" & "Treated"

Strain A fixed factor with 4 levels: "129Ola", "A/J", "NIH" & "BALB/C"

GST Numerical dependent variable indicating GST activity measurement `data_2w_Tdeath` *In vitro measurement of percentage cell death - two-way ANOVA design with repeated measures, and randomised blocks.*

Description

These are measurements of death of infected host cells (as percentage of total cells) upon infection with two strains of bacteria, measured at two time points, in 6 independent experiments. These data repeated-measures data suitable for two-way linear mixed effects modeling with experiment and subjects as random factors.

Usage

```
data_2w_Tdeath
```

Format

data.frame: 24 obs. of 6 variables:

Experiment A random factor with 6 levels "e1", "e2"...

Time A fixed factor with 2 levels: "t100" & "t300".

Time2 A numeric column that allows plotting data on a quantitative "Time" axis. The "Time" column has "factor" type values that should be used for the ANOVA..

Genotype A fixed factor with 2 levels that we want to compare "WT" & "KO".

Subject A random factor with 12 levels: "s1", "s2"... These are cell culture wells that were measured at two time points, and indicate "subjects" that underwent repeated-measures within each of 6 experiments. Subject IDs for WT and KO are unique and clearly indicate different wells.

PI Numerical dependent variable indicating propidium iodide dye uptake as a measure of cell death. These are percentage of dead cells out of total cells plated.

`data_cholesterol` *Hierarchical data from 25 subjects either treated or not at 5 hospitals - two-way ANOVA design with repeated measures.*

Description

An example dataset on measurements of blood cholesterol levels measured in 5 subjects measured before and after receiving a Drug. Five patients each were recruited at 5 hospitals (a-e), so that there are 25 different subjects (1-25) measured twice. Data are from Micro/Immuno Stats

Usage

```
data_cholesterol
```

Format

tibble: 30 obs. of 3 variables:

Hospital Factor with 5 levels (a-e), representing different hospitals where subjects were recruited.

Subject A factor with 25 levels denoting individuals on whom measurements were made twice.

Treatment A factor with 2 levels indicating when measurements were made, i.e. before and after drug.

Cholesterol Numerical dependent variable indicating measured doubling time in min.

`data_doubling_time` *Doubling time of E.coli measured by 10 students three independent times.*

Description

An example dataset showing measurements of *E. coli* doubling times (in min) measured by 10 different students in 3 independent experiments each. Note that Experiments are just called Exp1-Exp3 even though Exp1 of any of the students are not connected in anyway - this will confuse R! Data are from Micro/Immuno Stats

Usage

```
data_doubling_time
```

Format

tibble: 30 obs. of 3 variables:

Student Factor with 10 levels, representing different students.

Experiment A factor with 3 levels representing independent experiments.

Doubling_time Numerical dependent variable indicating measured doubling time in min.

`data_t_pdiff` *Matched data from two groups where difference between them is consistent.*

Description

An example dataset for paired difference Student's *t* test. These are bodyweight (Mass) in grams of same mice left untreated or treated, which are two groups to be compared. The data are in a longtable format, and the two groups are levels within the factor "Condition". The Subject column lists ID of matched mice that were measured without and with treatment. These data are from Sanchez-Garrido *et al*, Sci Signal, 2018, DOI: 10.1126/scisignal.aat6903.

Usage

```
data_t_pdiff
```

Format

data.frame: 20 obs. of 3 variables:

Subject Factor with 10 levels, denoted by capital letters, representing individuals or subjects.

Condition A fixed factor with 2 levels: "Untreated" & "Treated".

Mass Numerical dependent variable indicating body mass of mice

---

`data_t_pratio` *Matched data from two groups where ratio between them is consistent.*

Description

An example dataset for paired ratio Student's *t* test. These are Cytokine measurements by ELISA (in ng/ml) from 33 independent in vitro experiments performed on two Genotypes that we want to compare. The data are in a longtable format, and the two groups are levels within the factor "Genotype". The Experiment column lists ID of matched experiments.

Usage

```
data_t_pratio
```

Format

data.frame: 66 obs. of 3 variables:

Genotype Factor with 2 levels, representing genotypes to be compared ("WT" & "KO").

Experiment A random factor with 33 levels representing independent experiments, denoted as "Exp_1", "Exp_2"...

Cytokine Numerical dependent variable indicating cytokine measured by ELISA.

`get_graf_colours` *Get graf internal*

Description

Function to make grafify colour scheme. Thank you Dr Simon.

Usage

```
get_graf_colours(...)
```

Arguments

`...`	internal

Details

To visualise grafify colours use `plot_grafify_palette`.

Value

This function returns names and hexcodes of colours in grafify as a character vector.

`graf_colours` *List of hexcodes of colours in grafify palettes*

Description

To visualise these colours use `plot_grafify_palette`. `okabe_ito`, `bright`, `contrast`, `dark`, `light`, `muted`, `pale`, `vibrant`, `yello_conti` from Paul Tol's post. Zesty, Pastel, Elegant from this link. Colour hexcodes for `fishy`, `kelly`, `r4`, `safe`, `OrBl_div`, `PrGn_div`, `blue_conti`, `grey_conti` taken from cols4all:c4a_gui package. All schemes are colourblind-friendly.

Usage

```
graf_colours
```

Format

An object of class `character` of length 154.

Value

This is a character vector with names and hexcodes of colours used by palette functions. It is used by `get_graf_colours` to generate palettes.

`graf_col_palette` *Call palettes for scale & fill*

Description

Call palettes for scale & fill

Usage

```
graf_col_palette(palette = "okabe_ito", reverse = FALSE, ...)
```

Arguments

`palette`	internal
`reverse`	internal
`...`	additional parameters

Value

This generates required number of sequential colours from the chosen grafify palette when called by scale functions of ggplot2.

`graf_col_palette_default` *Call palettes for scale & fill with default colorRampPalette*

Description

Call palettes for scale & fill with default colorRampPalette

Usage

```
graf_col_palette_default(palette = "okabe_ito", reverse = FALSE, ...)
```

Arguments

`palette`	internal
`reverse`	internal
`...`	additional parameters

Value

This generates required number of distant colours from the chosen grafify palette when called by scale functions of ggplot2.

`graf_palettes` *List of palettes available in grafify package*

Description

To visualise these colours use `plot_grafify_palette`.

Usage

```
graf_palettes
```

Format

An object of class `list` of length 18.

Value

This function returns a list of palettes in grafify with names and hexcodes of colours in those palettes. Names of palettes available are as follows:

Categorical/discreet palettes:

- `okabe_ito`
- `bright`
- `contrast`
- `dark`
- `kelly`
- `light`
- `muted`
- `pale`
- `r4`
- `safe`
- `vibrant`

Sequential quantitative palettes:

- `grey_conti`
- `blue_conti`
- `yellow_conti`

Divergent quantitative palettes:

- `OrBl_div`
- `PrGn_div`

make_1way_data — *Make one-way or two-way independent group or randomised block design data.*

Description

The `make_1way_data`, `make_1way_rb_data`, `make_2way_data` and `make_2way_rb_data` functions generate independent or randomised block (rb) design data of one-way or two-way designs.

Usage

```
make_1way_data(Group_means, Num_obs, Residual_SD)
```

Arguments

`Group_means`	a vector with means of each level of the first fixed factor (FixFac_X1) measured within Group 1.
`Num_obs`	a single numeric value indicating the number of independent measurements, i.e. levels within the random factor Experiment.
`Residual_SD`	a single numeric value indicating residual SD in the model.

Details

Random variates from the normal distribution based on user provided mean and SD provided are generated. For independent designs, the `Residual_SD` argument is used to set expected residual SD from the linear model. Exp_SD is used to set experiment-to-experiment SD, that will be assigned to the random factor for rb designs.

Num_exp sets the number of independent measurements per group.

For one-way designs, the user provides Group_means as a vector. Number of levels are recognised based on number of means. For two-way designs, two vectors are to be provided by the user containing means of levels of a second factor. Number of means in both vectors should be the same. These functions can only handle balanced designs, i.e. same number of observations in all groups.

The output is a data frame with one or two columns denoting the fixed factor with levels that match the number of means entered. For rb data, the column for RandFac denotes levels of the blocking factor. The quantitative response variables are in the numeric Values column.

Value

This function produces a `data.frame` object containing simulated data.

Examples

```
#Basic usage with three levels within Factor_X,
#20 observations in each group, with residual SD 15

one_independent_tab <- make_1way_data(c(350, 250, 100), 15, 20)

str(one_independent_tab)
head(one_independent_tab)
```

---

make_1way_rb_data   *Make one-way or two-way independent group or randomised block design data.*

Description

The make_1way_data, make_1way_rb_data, make_2way_data and make_2way_rb_data functions generate independent or randomised block (rb) design data of one-way or two-way designs.

Usage

```
make_1way_rb_data(Group_means, Num_exp, Exp_SD, Residual_SD)
```

Arguments

Group_means	a vector with means of each level of the first fixed factor (FixFac_X1) measured within Group 1.
Num_exp	a single numeric value. indicating the number of independent measurements, i.e. levels within the random factor RandFac.
Exp_SD	a single numeric value indicating the standard deviation (SD) between experiments, i.e. within RandFac.
Residual_SD	a single numeric value indicating residual SD in the model.

Details

Random variates from the normal distribution based on user provided mean and SD provided are generated. For independent designs, the Residual_SD argument is used to set expected residual SD from the linear model. Exp_SD is used to set experiment-to-experiment SD, that will be assigned to the random factor for rb designs.

Num_exp sets the number of independent measurements per group.

For one-way designs, the user provides Group_means as a vector. Number of levels are recognised based on number of means. For two-way designs, two vectors are to be provided by the user containing means of levels of a second factor. Number of means in both vectors should be the same. These functions can only handle balanced designs, i.e. same number of observations in all groups.

The output is a data frame with one or two columns denoting the fixed factor with levels that match the number of means entered. For rb data, the column for RandFac denotes levels of the blocking factor. The quantitative response variables are in the numeric Values column.

Value

This function produces a `data.frame` object containing simulated data.

Examples

```
#Basic usage with two levels within FactorX2,
#20 experiments with inter-experiment SD 20, and residual SD 15

two_rb_tab <- make_2way_rb_data(c(100, 20), c(200, 300), 20, 20, 15)

str(two_rb_tab)
head(two_rb_tab)
```

`make_2way_data` *Make one-way or two-way independent group or randomised block design data.*

Description

The `make_1way_data`, `make_1way_rb_data`, `make_2way_data` and `make_2way_rb_data` functions generate independent or randomised block (rb) design data of one-way or two-way designs.

Usage

```
make_2way_data(Group_1_means, Group_2_means, Num_obs, Residual_SD)
```

Arguments

`Group_1_means`	a vector with means of each level of the first fixed factor (FixFac_X1) measured within Group 1.
`Group_2_means`	only for `make_2way_data` and `make_2way_rb_data`: a vector with mean(s) of each level of FactorX2 measured within Group 2.
`Num_obs`	a single numeric value indicating the number of independent measurements, i.e. levels within the random factor Experiment.
`Residual_SD`	a single numeric value indicating residual SD in the model.

Details

Random variates from the normal distribution based on user provided mean and SD provided are generated. For independent designs, the `Residual_SD` argument is used to set expected residual SD from the linear model. Exp_SD is used to set experiment-to-experiment SD, that will be assigned to the random factor for rb designs.

Num_obs sets the number of independent measurements per group.

For one-way designs, the user provides Group_means as a vector. Number of levels are recognised based on number of means. For two-way designs, two vectors are to be provided by the user containing means of levels of a second factor. Number of means in both vectors should be the same. These functions can only handle balanced designs, i.e. same number of observations in all groups.

The output is a data frame with one or two columns denoting the fixed factor with levels that match the number of means entered. For rb data, the column for RandFac denotes levels of the blocking factor. The quantitative response variables are in the numeric Values column.

Value

This function produces a `data.frame` object containing simulated data.

Examples

```
#Basic usage with two levels within FactorX2, 20 observations in each group, with residual SD 15

two_independent_tab <- make_2way_data(c(100, 20), c(200, 300), 20, 15)

#Four levels with 5 observations and residual SD 5
two_independent_tab <- make_2way_data(c(100, 20, 1500, 20), c(150, 5, 1450, 25), 5, 5)
```

`make_2way_rb_data` *Make one-way or two-way independent group or randomised block design data.*

Description

The `make_1way_data`, `make_1way_rb_data`, `make_2way_data` and `make_2way_rb_data` functions generate independent or randomised block (rb) design data of one-way or two-way designs.

Usage

```
make_2way_rb_data(Group1_means, Group2_means, Num_exp, Exp_SD, Residual_SD)
```

Arguments

`Group1_means`	a vector with means of each level of the first fixed factor (FixFac_X1) measured within Group 1.
`Group2_means`	only for `make_2way_data` and `make_2way_rb_data`: a vector with mean(s) of each level of FactorX2 measured within Group 2.
`Num_exp`	a single numeric value indicating the number of independent measurements, i.e. levels within the random factor RandFac.
`Exp_SD`	a single numeric value indicating the standard deviation (SD) between experiment, i.e. within RandFac.
`Residual_SD`	a single numeric value indicating residual SD in the model.

Details

Random variates from the normal distribution based on user provided mean and SD provided are generated. For independent designs, the `Residual_SD` argument is used to set expected residual SD from the linear model. Exp_SD is used to set experiment-to-experiment SD, that will be assigned to the random factor (RandFac) for rb designs.

Num_exp sets the number of independent measurements per group.

For one-way designs, the user provides Group_means as a vector. Number of levels are recognised based on number of means. For two-way designs, two vectors are to be provided by the user containing means of levels of a second factor. Number of means in both vectors should be the same. These functions can only handle balanced designs, i.e. same number of observations in all groups.

The output is a data frame with one or two columns denoting the fixed factor with levels that match the number of means entered. For rb data, the column for RandFac denotes levels of the blocking factor. The quantitative response variables are in the numeric Values column.

Value

This function produces a `data.frame` object containing simulated data.

Examples

```
#Basic usage with two levels within FactorX2,
#20 experiments with inter-experiment SD 20, and residual SD 15

two_rb_tab <- make_2way_rb_data(c(100, 20), c(200, 300), 20, 20, 15)

str(two_rb_tab)
head(two_rb_tab)
```

`mixed_anova` *ANOVA table from linear mixed effects analysis.*

Description

There are four related functions for mixed effects analyses: `mixed_model`, `mixed_anova`, `mixed_model_slopes`, and `mixed_anova_slopes`.

Usage

```
mixed_anova(
  data,
  Y_value,
  Fixed_Factor,
  Random_Factor,
  Df_method = "Kenward-Roger",
  SS_method = "II",
  ...
)
```

Arguments

`data`	a data table object, e.g. data.frame or tibble.
`Y_value`	name of column containing quantitative (dependent) variable, provided within "quotes".
`Fixed_Factor`	name(s) of categorical fixed factors (independent variables) provided as a vector if more than one or within "quotes".
`Random_Factor`	name(s) of random factors to allow random intercepts; to be provided as a vector when more than one or within "quotes".
`Df_method`	method for calculating degrees of freedom. Default is Kenward-Roger, can be changed to "Satterthwaite".
`SS_method`	type of sum of square, default is type II, can be changed to "I", "III", "1" or "2", or others.
`...`	any additional arguments to pass on to `lmer` if required.

Details

This function uses `lmer` to fit a linear mixed effect model and provides the model object, which could be used for post-hoc comparisons. The model object is converted to class `lmerModLmerTest` object by `as_lmerModLmerTest`. This is then passes on the model to `anova` and provides the ANOVA table with F and P values. It produces a type II sum of squares ANOVA table with Kenward-Roger approximation for degrees of freedom (as implemented in `lmerTest`) package. It requires a data table, one dependent variable (Y_value), one or more independent variables (Fixed_Factor), and at least one random factor (Random_Factor). These should match names of variables in the long-format data table exactly. This function is related to `mixed_model`.

More than one fixed factors can be provided as a vector (e.g. c("A", "B")). A full model with interaction term is fitted. This means when `Y_value = Y`, `Fixed_factor = c("A", "B")`, `Random_factor = "R"` are entered as arguments, these are passed on as `Y ~ A*B + (1|R)` (which is equivalent to `Y ~ A + B + A:B + (1|R)`). For simplicity, only random intercepts are fitted (`(1|R)`).

Value

ANOVA table of class "anova" and "data.frame".

Examples

```
#Usage with one fixed (Student) and random factor (Experiment)
mixed_anova(data = data_doubling_time,
Y_value = "Doubling_time",
Fixed_Factor = "Student",
Random_Factor = "Experiment")

#two fixed factors provided as a vector
mixed_anova(data = data_cholesterol,
Y_value = "Cholesterol",
Fixed_Factor = c("Treatment", "Hospital"),
Random_Factor = "Subject")
```

mixed_anova_slopes — *ANOVA table from linear mixed effects analysis.*

Description

There are four related functions for mixed effects analyses: `mixed_model`, `mixed_anova`, `mixed_model_slopes`, and `mixed_anova_slopes`.

Usage

```
mixed_anova_slopes(
  data,
  Y_value,
  Fixed_Factor,
  Slopes_Factor,
  Random_Factor,
  Df_method = "Kenward-Roger",
  SS_method = "II",
  ...
)
```

Arguments

`data`	a data table object, e.g. data.frame or tibble.
`Y_value`	name of column containing quantitative (dependent) variable, provided within "quotes".
`Fixed_Factor`	name(s) of categorical fixed factors (independent variables) provided as a vector if more than one or within "quotes".
`Slopes_Factor`	name of factor to allow varying slopes on.
`Random_Factor`	name(s) of random factors to allow random intercepts; to be provided as a vector when more than one or within "quotes".
`Df_method`	method for calculating degrees of freedom. Default is Kenward-Roger, can be changed to "Satterthwaite".
`SS_method`	type of sum of square, default is type II, can be changed to "I", "III", "1" or "2", or others.
`...`	any additional arguments to pass on to `lmer` if required.

Details

This function uses `lmer` to fit a linear mixed effect model and provides the model object, which could be used for post-hoc comparisons. The model object is converted to class `lmerModLmerTest` object by `as_lmerModLmerTest`.

It produces a type II sum of squares ANOVA table with Kenward-Roger approximation for degrees of freedom (as implemented in `lmerTest`) package. It requires a data table, one dependent variable (Y_value), one or more independent variables (Fixed_Factor). Exactly one random factor (Random_Factor) and Slope_Factor should be provided. This function is related to `mixed_model`.

More than one fixed factors can be provided as a vector (e.g. c("A", "B")). A full model with interaction term is fitted with one term each for varying slopes and intercepts. This means when `Y_value = Y, Fixed_factor = c("A", "B"), Slopes_Factor = "S", Random_factor = "R"` are entered as arguments, these are passed on as `Y ~ A*B + (S|R)` (which is equivalent to `Y ~ A + B + A:B + (S|R)`). In this experimental implementation, random slopes and intercepts are fitted (`(Slopes_Factor|Random_Factor)`). Only one term each is allowed for ¬ and `Random_Factor`.

Value

ANOVA table of class "anova" and "data.frame".

Examples

```
mixed_anova_slopes(data = data_2w_Tdeath,
Y_value = "PI",
Fixed_Factor = c("Genotype", "Time"),
Slopes_Factor = "Time",
Random_Factor = "Experiment")
```

`mixed_model` *Model from a linear mixed effects model*

Description

There are four related functions for mixed effects analyses: `mixed_model`, `mixed_anova`, `mixed_model_slopes`, and `mixed_anova_slopes`.

Usage

```
mixed_model(data, Y_value, Fixed_Factor, Random_Factor, ...)
```

Arguments

`data`	a data table object, e.g. data.frame or tibble.
`Y_value`	name of column containing quantitative (dependent) variable, provided within "quotes".
`Fixed_Factor`	name(s) of categorical fixed factors (independent variables) provided as a vector if more than one or within "quotes".
`Random_Factor`	name(s) of random factors to allow random intercepts; to be provided as a vector when more than one or within "quotes".
`...`	any additional arguments to pass on to `lmer` if required.

Details

This function uses `lmer` to fit a linear mixed effect model and provides the model object, which could be used for post-hoc comparisons. The model object is converted to class `lmerModLmerTest` object by `as_lmerModLmerTest`.

It requires a data table, one dependent variable (Y_value), one or more independent variables (Fixed_Factor), and at least one random factor (Random_Factor). These should match names of variables in the long-format data table exactly. This function is related to `mixed_anova`. Output of this function can be used with `posthoc_Pairwise`, `posthoc_Levelwise` and `posthoc_vsRef`, or with `emmeans`.

More than one fixed factors can be provided as a vector (e.g. c("A", "B")). A full model with interaction term is fitted. This means when `Y_value = Y`, `Fixed_factor = c("A", "B")`, `Random_factor = "R"` are entered as arguments, these are passed on as `Y ~ A*B + (1|R)` (which is equivalent to `Y ~ A + B + A:B + (1|R)`). For simplicity, only random intercepts are fitted (`(1|R)`).

Also see `mixed_anova_slopes` and `mixed_model_slopes` for similar functions where variable slopes and intercept models are fit.

Value

This function returns an S4 object of class "lmerModLmerTest".

Examples

```
#one fixed factor and random factor
mixed_model(data = data_doubling_time,
Y_value = "Doubling_time",
Fixed_Factor = "Student",
Random_Factor = "Experiment")

#two fixed factors as a vector, one random factor
mixed_model(data = data_cholesterol,
Y_value = "Cholesterol",
Fixed_Factor = c("Treatment", "Hospital"),
Random_Factor = "Subject")

#save model
model <- mixed_model(data = data_doubling_time,
Y_value =  "Doubling_time",
Fixed_Factor = "Student",
Random_Factor = "Experiment")

#get model summary
summary(model)
```

`mixed_model_slopes` *Model from a linear mixed effects model with varying slopes*

Description

There are four related functions for mixed effects analyses: `mixed_model`, `mixed_anova`, `mixed_model_slopes`, and `mixed_anova_slopes`.

Usage

```
mixed_model_slopes(
  data,
  Y_value,
  Fixed_Factor,
  Slopes_Factor,
  Random_Factor,
  ...
)
```

Arguments

`data`	a data table object, e.g. data.frame or tibble.
`Y_value`	name of column containing quantitative (dependent) variable, provided within "quotes".
`Fixed_Factor`	name(s) of categorical fixed factors (independent variables) provided as a vector if more than one or within "quotes".
`Slopes_Factor`	name of factor to allow varying slopes on.
`Random_Factor`	name(s) of random factors to allow random intercepts; to be provided as a vector when more than one or within "quotes".
`...`	any additional arguments to pass on to `lmer` if required.

Details

This function uses `lmer` to fit a linear mixed effect model and provides the model object, which could be used for post-hoc comparisons. The model object is converted to class `lmerModLmerTest` object by `as_lmerModLmerTest`. It requires a data table, one dependent variable (Y_value), one or more independent variables (Fixed_Factor). Exactly one random factor (Random_Factor) and Slope_Factor should be provided. This function is related to `mixed_anova_slopes`. Output of this function can be used with `posthoc_Pairwise`, `posthoc_Levelwise` and `posthoc_vsRef`, or with `emmeans`.

More than one fixed factors can be provided as a vector (e.g. c("A", "B")). A full model with interaction term is fitted with one term each for varying slopes and intercepts. This means when `Y_value = Y, Fixed_factor = c("A", "B"), Slopes_Factor = "S", Random_factor = "R"` are entered as arguments, these are passed on as `Y ~ A*B + (S|R)` (which is equivalent to `Y ~ A + B + A:B + (S|R)`). In this experimental implementation, random slopes and intercepts are fitted (`(Slopes_Factor|Random_Factor)`). Only one term each is allowed for `Slopes_Factor` and `Random_Factor`.

Value

This function returns an S4 object of class "lmerModLmerTest".

Examples

```
#two fixed factors as a vector,
#exactly one slope factor and random factor
mod <- mixed_model_slopes(data = data_2w_Tdeath,
Y_value = "PI",
Fixed_Factor = c("Genotype", "Time"),
Slopes_Factor = "Time",
Random_Factor = "Experiment")
#get summary
summary(mod)
```

`plot_3d_scatterbar` *Plot a scatter graph with matched shapes on a bar plot using three variables.*

Description

The functions `plot_3d_scatterbar`, `plot_3d_scatterbox`, `plot_4d_scatterbar` and `plot_4d_scatterbox` are useful for plotting one-way or two-way ANOVA designs with randomised blocks or repeated measures. The blocks or subjects can be mapped to the shapes argument in both functions (up to 25 levels can be mapped to shapes; there will be an error if this number is exceeded). The 3d versions use the categorical variable (`xcol`) for grouping (e.g. one-way ANOVA designs), and 4d versions take an additional grouping variable (e.g. two-way ANOVA designs) that is passed to either `boxes` or `bars` argument.

Usage

```
plot_3d_scatterbar(
  data,
  xcol,
  ycol,
  shapes,
  ewid = 0.2,
  symsize = 2.5,
  symthick = 1,
  jitter = 0.2,
  fontsize = 20,
  b_alpha = 1,
  s_alpha = 1,
  ColSeq = TRUE,
  ColPal = c("okabe_ito", "all_grafify", "bright", "contrast", "dark", "fishy",
    "kelly", "light", "muted", "pale", "r4", "safe", "vibrant"),
  ColRev = FALSE,
  SingleColour = "NULL",
  TextXAngle = 0,
  ...
)
```

Arguments

data	a data table, e.g. data.frame or tibble.
xcol	name of the column with the categorical factor to be plotted on X axis.
ycol	name of the column with quantitative variable to plot on the Y axis.
shapes	name of the column with the second categorical factor, for example from a two-way ANOVA design.
ewid	width of error bars, default set to 0.2.
symsize	size of symbols, default set to 3.
symthick	size of outline of symbol lines (`stroke = 1.5`), default set to 1.5
jitter	extent of jitter (scatter) of symbols, default is 0.2. Increase to reduce symbol overlap, set to 0 for aligned symbols.
fontsize	parameter of `base_size` of fonts in `theme_classic`, default set to size 20.
b_alpha	fractional opacity of bars, default set to 1 (i.e. maximum opacity & zero transparency).
s_alpha	fractional opacity of symbols, default set to 1 (i.e. maximum opacity & zero transparency).
ColSeq	logical TRUE or FALSE. Default TRUE for sequential colours from chosen palette. Set to FALSE for distant colours, which will be applied using `scale_fill_grafify2`.
ColPal	grafify colour palette to apply, default "okabe_ito"; see `graf_palettes` for available palettes.
ColRev	whether to reverse order of colour within the selected palette, default F (FALSE); can be set to T (TRUE).
SingleColour	a colour hexcode (starting with #), a number between 1-154, or names of colours from `grafify` colour palettes to fill along X-axis aesthetic. Accepts any colour other than "black"; use `grey_lin11`, which is almost black.
TextXAngle	orientation of text on X-axis; default 0 degrees. Change to 45 or 90 to remove overlapping text.
...	any additional arguments to pass to `ggplot2`geom_boxplot or `ggplot2`geom_point.

Details

These functions rely on `ggplot` with `geom_point` and `geom_bar` (through `stat_summary`) or `geom_boxplot` geometries.

Variables other than the quantitative variable (`ycol`) will be automatically converted to categorical variables even if they are numeric in the data table.

Shapes are always plotted in black colour, and their opacity can be changed with the `s_alpha` argument and overlap can be reduced with the `jitter` argument. Other arguments are similar to other plot functions as briefly explained below.

Bars depict means using `stat_summary` with `geom = "bar", fun = "mean"` , and bar width is set to 0.7 (cannot be changed). Error bar width can be changed with the `ewid` argument.

Boxplot geometry uses `geom_boxplot` with `position = position_dodge(width = 0.9), width = 0.6`. The thick line within the boxplot depicts the median, the box the IQR (interquantile range) and the whiskers show 1.5*IQR.

In 4d versions, the two grouping variables (i.e. `xcol` and either `boxes` or `bars`) are passed to ggplot aesthetics through `group = interaction{ xcol, shapes}`.

Colours can be changed using `ColPal`, `ColRev` or `ColSeq` arguments. `ColPal` can be one of the following: "okabe_ito", "dark", "light", "bright", "pale", "vibrant, "muted" or "contrast". `ColRev` (logical TRUE/FALSE) decides whether colours are chosen from first-to-last or last-to-first from within the chosen palette. `ColSeq` (logical TRUE/FALSE) decides whether colours are picked by respecting the order in the palette or the most distant ones using `colorRampPalette`.

All four functions can be expanded further, for example with `facet_grid` or `facet_wrap`.

Value

This function returns a `ggplot2` object of class "gg" and "ggplot".

Examples

```
#3d version for 1-way data with blocking
plot_3d_scatterbar(data = data_1w_death,
xcol = Genotype, ycol = Death,
shapes = Experiment)
#compare above graph to
plot_scatterbar_sd(data = data_1w_death,
xcol = Genotype, ycol = Death)
#single colour
plot_3d_scatterbar(data = data_1w_death,
xcol = Genotype, ycol = Death,
shapes = Experiment,
SingleColour = "pale_grey")

#4d version for 2-way data with blocking
plot_4d_scatterbox(data = data_2w_Tdeath,
xcol = Genotype,
ycol = PI,
boxes = Time,
shapes = Experiment)

plot_4d_scatterbar(data = data_2w_Festing,
xcol = Strain,
ycol = GST,
bars = Treatment,
shapes = Block)
```

plot_3d_scatterbox *Plot a scatter and box plot with matched symbols.*

Description

The functions plot_3d_scatterbar, plot_3d_scatterbox, plot_4d_scatterbar and plot_4d_scatterbox are useful for plotting one-way or two-way ANOVA designs with randomised blocks or repeated measures. The blocks or subjects can be mapped to the shapes argument in both functions (up to 25 levels can be mapped to shapes; there will be an error if this number is exceeded). The 3d versions use the categorical variable (xcol) for grouping (e.g. one-way ANOVA designs), and 4d versions take an additional grouping variable (e.g. two-way ANOVA designs) that is passed to either boxes or bars argument.

Usage

```
plot_3d_scatterbox(
  data,
  xcol,
  ycol,
  shapes,
  symsize = 2.5,
  symthick = 1,
  jitter = 0.2,
  fontsize = 20,
  b_alpha = 1,
  s_alpha = 1,
  ColSeq = TRUE,
  ColPal = c("okabe_ito", "all_grafify", "bright", "contrast", "dark", "fishy",
    "kelly", "light", "muted", "pale", "r4", "safe", "vibrant"),
  ColRev = FALSE,
  SingleColour = "NULL",
  TextXAngle = 0,
  ...
)
```

Arguments

data	a data table, e.g. data.frame or tibble.
xcol	name of the column with the categorical factor to be plotted on X axis. If your table has numeric X, enter xcol = factor(name of colum).
ycol	name of the column with quantitative variable to plot on the Y axis.
shapes	name of the column with the second categorical factor in a two-way ANOVA design.
symsize	size of symbols, default set to 3.
symthick	size of outline of symbol lines (stroke = 1.0), default set to 1.0.
jitter	extent of jitter (scatter) of symbols, default is 0.2. Increase to reduce symbol overlap, set to 0 for aligned symbols.
fontsize	parameter of base_size of fonts in theme_classic, default set to size 20.
b_alpha	fractional opacity of boxes, default set to 1 (i.e. maximum opacity & zero transparency).

`s_alpha`	fractional opacity of symbols, default set to 1 (i.e. maximum opacity & zero transparency).
`ColSeq`	logical TRUE or FALSE. Default TRUE for sequential colours from chosen palette. Set to FALSE for distant colours, which will be applied using `scale_fill_grafify2`.
`ColPal`	grafify colour palette to apply, default "okabe_ito"; see `graf_palettes` for available palettes.
`ColRev`	whether to reverse order of colour within the selected palette, default F (FALSE); can be set to T (TRUE).
`SingleColour`	a colour hexcode (starting with #), a number between 1-154, or names of colours from `grafify` colour palettes to fill along X-axis aesthetic. Accepts any colour other than "black"; use `grey_lin11`, which is almost black.
`TextXAngle`	orientation of text on X-axis; default 0 degrees. Change to 45 or 90 to remove overlapping text.
`...`	any additional arguments to pass to `ggplot2`geom_boxplot.

Details

These functions rely on `ggplot` with `geom_point` and `geom_bar` (through `stat_summary`) or `geom_boxplot` geometries.

Variables other than the quantitative variable (`ycol`) will be automatically converted to categorical variables even if they are numeric in the data table.

Shapes are always plotted in black colour, and their opacity can be changed with the `s_alpha` argument and overlap can be reduced with the `jitter` argument. Other arguments are similar to other plot functions as briefly explained below.

Bars depict means using `stat_summary` with `geom = "bar", fun = "mean"` , and bar width is set to 0.7 (cannot be changed). Error bar width can be changed with the `ewid` argument.

Boxplot geometry uses `geom_boxplot` with `position = position_dodge(width = 0.9), width = 0.6`. The thick line within the boxplot depicts the median, the box the IQR (interquantile range) and the whiskers show 1.5*IQR.

In 4d versions, the two grouping variables (i.e. `xcol` and either `boxes` or `bars`) are passed to ggplot aesthetics through `group = interaction{ xcol, shapes}`.

Colours can be changed using `ColPal`, `ColRev` or `ColSeq` arguments. `ColPal` can be one of the following: "okabe_ito", "dark", "light", "bright", "pale", "vibrant, "muted" or "contrast". `ColRev` (logical TRUE/FALSE) decides whether colours are chosen from first-to-last or last-to-first from within the chosen palette. `ColSeq` (logical TRUE/FALSE) decides whether colours are picked by respecting the order in the palette or the most distant ones using `colorRampPalette`.

All four functions can be expanded further, for example with `facet_grid` or `facet_wrap`.

Value

This function returns a `ggplot2` object of class "gg" and "ggplot".

Examples

```
#3d version for 1-way data with blocking
plot_3d_scatterbox(data = data_1w_death,
xcol = Genotype, ycol = Death,
shapes = Experiment)
#compare above graph to
plot_scatterbox(data = data_1w_death,
xcol = Genotype, ycol = Death)
#single colour graph
plot_3d_scatterbox(data = data_1w_death,
xcol = Genotype, ycol = Death,
shapes = Experiment,
SingleColour = "pale_grey")

#4d version for 2-way data with blocking
plot_4d_scatterbox(data = data_2w_Tdeath,
xcol = Genotype,
ycol = PI,
boxes = Time,
shapes = Experiment)

plot_4d_scatterbar(data = data_2w_Festing,
xcol = Strain,
ycol = GST,
bars = Treatment,
shapes = Block)
```

`plot_3d_scatterviolin` *Plot a scatter with violin & box plot with matched symbols.*

Description

The functions `plot_3d_scatterbar`, `plot_3d_scatterbox`, `plot_3d_scatterviolin`, `plot_4d_scatterbar`, `plot_4d_scatterbox` and `plot_4d_scatterviolin` are useful for plotting one-way or two-way ANOVA designs with randomised blocks or repeated measures. The blocks or subjects can be mapped to the shapes argument in both functions (up to 25 levels can be mapped to shapes; there will be an error if this number is exceeded). The 3d versions use the categorical variable (`xcol`) for grouping (e.g. one-way ANOVA designs), and 4d versions take an additional grouping variable (e.g. two-way ANOVA designs) that is passed to either `boxes` or `bars` argument.

Usage

```
plot_3d_scatterviolin(
  data,
  xcol,
  ycol,
  shapes,
  symsize = 2.5,
  s_alpha = 1,
  symthick = 1,
  v_alpha = 1,
  b_alpha = 0,
  bwid = 0.2,
  bvthick = 1,
  jitter = 0.2,
  fontsize = 20,
  ColSeq = TRUE,
  ColPal = c("okabe_ito", "all_grafify", "bright", "contrast", "dark", "fishy",
    "kelly", "light", "muted", "pale", "r4", "safe", "vibrant"),
  ColRev = FALSE,
  SingleColour = "NULL",
  TextXAngle = 0,
  scale = "width",
  trim = TRUE,
  ...
)
```

Arguments

`data`	a data table, e.g. data.frame or tibble.
`xcol`	name of the column with the categorical factor to be plotted on X axis. If your table has numeric X, enter `xcol = factor(name of colum)`.
`ycol`	name of the column with quantitative variable to plot on the Y axis.
`shapes`	name of the column with the second categorical factor in a two-way ANOVA design.
`symsize`	size of symbols, default set to 3.
`s_alpha`	fractional opacity of symbols, default set to 1 (i.e. maximum opacity & zero transparency). Set `s_alpha = 0` to not show scatter plot.
`symthick`	size of outline of symbol lines (`stroke = 1.0`), default set to 1.0.
`v_alpha`	fractional opacity of violins, default set to 1 (i.e. maximum opacity & zero transparency).
`b_alpha`	fractional opacity of boxplots. Default is set to 0, which results in white boxes inside violins. Change to any value >0 up to 1 for different levels of transparency.
`bwid`	width of boxplots; default 0.2
`bvthick`	thickness of both violin and boxplot lines; default 1.
`jitter`	extent of jitter (scatter) of symbols, default is 0 (i.e. aligned symbols). To reduce symbol overlap, try 0.1-0.3 or higher.
`fontsize`	parameter of `base_size` of fonts in `theme_classic`, default set to size 20.
`ColSeq`	logical TRUE or FALSE. Default TRUE for sequential colours from chosen palette. Set to FALSE for distant colours, which will be applied using `scale_fill_grafify2`.
`ColPal`	grafify colour palette to apply, default "okabe_ito"; see `graf_palettes` for available palettes.

`ColRev`	whether to reverse order of colour within the selected palette, default F (FALSE); can be set to T (TRUE).
`SingleColour`	a colour hexcode (starting with #), a number between 1-154, or names of colours from `grafify` colour palettes to fill along X-axis aesthetic. Accepts any colour other than "black"; use `grey_lin11`, which is almost black.
`TextXAngle`	orientation of text on X-axis; default 0 degrees. Change to 45 or 90 to remove overlapping text.
`scale`	set to "area" by default, can be changed to "count" or "width".
`trim`	set whether tips of violin plot should be trimmed at high/low data. Default `trim = T`, can be changed to F.
`...`	any additional arguments to pass to `ggplot2`geom_boxplot or `ggplot2`geom_violin.

Details

These functions rely on `ggplot` with `geom_point` and `geom_bar` (through `stat_summary`), or `geom_violin` and `geom_boxplot` geometries.

Variables other than the quantitative variable (`ycol`) will be automatically converted to categorical variables even if they are numeric in the data table.

Shapes are always plotted in black colour, and their opacity can be changed with the `s_alpha` argument and overlap can be reduced with the `jitter` argument. Other arguments are similar to other plot functions as briefly explained below.

Bars depict means using `stat_summary` with `geom = "bar", fun = "mean"` , and bar width is set to 0.7 (cannot be changed). Error bar width can be changed with the `ewid` argument.

Boxplot geometry uses `geom_boxplot` with `position = position_dodge(width = 0.9), width = 0.6`. The thick line within the boxplot depicts the median, the box the IQR (interquantile range) and the whiskers show 1.5*IQR.

In 4d versions, the two grouping variables (i.e. `xcol` and either `boxes` or `bars`) are passed to ggplot aesthetics through `group = interaction{ xcol, shapes}`.

Colours can be changed using `ColPal`, `ColRev` or `ColSeq` arguments. `ColPal` can be one of the following: "okabe_ito", "dark", "light", "bright", "pale", "vibrant, "muted" or "contrast". `ColRev` (logical TRUE/FALSE) decides whether colours are chosen from first-to-last or last-to-first from within the chosen palette. `ColSeq` (logical TRUE/FALSE) decides whether colours are picked by respecting the order in the palette or the most distant ones using `colorRampPalette`.

All four functions can be expanded further, for example with `facet_grid` or `facet_wrap`.

Value

This function returns a `ggplot2` object of class "gg" and "ggplot".

Examples

```
#3d version for 1-way data with blocking
plot_3d_scatterviolin(data = data_1w_death,
xcol = Genotype, ycol = Death,
shapes = Experiment)
#compare above graph to
```

```
plot_scatterviolin(data = data_1w_death,
xcol = Genotype, ycol = Death)
#single colour
plot_3d_scatterviolin(data = data_1w_death,
xcol = Genotype, ycol = Death,
shapes = Experiment,
SingleColour = "pale_grey")

#4d version for 2-way data with blocking
plot_4d_scatterviolin(data = data_2w_Tdeath,
xcol = Genotype,
ycol = PI,
boxes = Time,
shapes = Experiment)
```

plot_4d_scatterbar *Plot a dot plot with matched shapes on a box plot using four variables.*

Description

The functions plot_3d_scatterbar, plot_3d_scatterbox, plot_4d_scatterbar and plot_4d_scatterbox are useful for plotting one-way or two-way ANOVA designs with randomised blocks or repeated measures. The blocks or subjects can be mapped to the shapes argument in both functions (up to 25 levels can be mapped to shapes; there will be an error if this number is exceeded). The 3d versions use the categorical variable (xcol) for grouping (e.g. one-way ANOVA designs), and 4d versions take an additional grouping variable (e.g. two-way ANOVA designs) that is passed to either boxes or bars argument.

Usage

```
plot_4d_scatterbar(
  data,
  xcol,
  ycol,
  bars,
  shapes,
  symsize = 2.5,
  symthick = 1,
  jitter = 0.2,
  ewid = 0.2,
  fontsize = 20,
  b_alpha = 1,
  s_alpha = 1,
  ColPal = c("okabe_ito", "all_grafify", "bright", "contrast", "dark", "fishy",
    "kelly", "light", "muted", "pale", "r4", "safe", "vibrant"),
  ColRev = FALSE,
  ColSeq = TRUE,
  TextXAngle = 0,
  ...
)
```

Arguments

`data`	a data table, e.g. data.frame or tibble.
`xcol`	name of the column with the categorical factor to plot on X axis. If column is numeric, enter as `factor(col)`.
`ycol`	name of the column to plot on quantitative variable on the Y axis.
`bars`	name of the column containing grouping within the factor plotted on X axis. Can be categorical or numeric X. If your table has numeric X and you want to plot as factor, enter `xcol` = `factor(name of colum)`.
`shapes`	name of the column that contains matched observations, e.g. subject IDs, experiment ID.
`symsize`	size of symbols, default set to 3.
`symthick`	size of outline of symbol lines (`stroke` = `1.0`), default set to 1.0
`jitter`	extent of jitter (scatter) of symbols, default is 0.2. Increase to reduce symbol overlap, set to 0 for aligned symbols.
`ewid`	width of error bars, default set to 0.2.
`fontsize`	parameter of `base_size` of fonts in `theme_classic`, default set to size 20.
`b_alpha`	fractional opacity of bars, default set to 1 (i.e. maximum opacity & zero transparency).
`s_alpha`	fractional opacity of symbols, default set to 1 (i.e. maximum opacity & zero transparency).
`ColPal`	grafify colour palette to apply, default "okabe_ito"; see `graf_palettes` for available palettes.
`ColRev`	whether to reverse order of colour within the selected palette, default F (FALSE); can be set to T (TRUE).
`ColSeq`	logical TRUE or FALSE. Default TRUE for sequential colours from chosen palette. Set to FALSE for distant colours, which will be applied using `scale_fill_grafify2`.
`TextXAngle`	orientation of text on X-axis; default 0 degrees. Change to 45 or 90 to remove overlapping text.
`...`	any additional arguments to pass to `ggplot2`stat_summary or `ggplot2`geom_point.

Details

These functions rely on `ggplot` with `geom_point` and `geom_bar` (through `stat_summary`) or `geom_boxplot` geometries.

Variables other than the quantitative variable (`ycol`) will be automatically converted to categorical variables even if they are numeric in the data table.

Shapes are always plotted in black colour, and their opacity can be changed with the `s_alpha` argument and overlap can be reduced with the `jitter` argument. Other arguments are similar to other plot functions as briefly explained below.

Bars depict means using stat_summary with geom = "bar", fun = "mean" , and bar width is set to 0.7 (cannot be changed). Error bar width can be changed with the ewid argument.

Boxplot geometry uses geom_boxplot with position = position_dodge(width = 0.9), width = 0.6. The thick line within the boxplot depicts the median, the box the IQR (interquantile range) and the whiskers show 1.5*IQR.

In 4d versions, the two grouping variables (i.e. xcol and either boxes or bars) are passed to ggplot aesthetics through group = interaction{ xcol, shapes}.

Colours can be changed using ColPal, ColRev or ColSeq arguments. ColPal can be one of the following: "okabe_ito", "dark", "light", "bright", "pale", "vibrant, "muted" or "contrast". ColRev (logical TRUE/FALSE) decides whether colours are chosen from first-to-last or last-to-first from within the chosen palette. ColSeq (logical TRUE/FALSE) decides whether colours are picked by respecting the order in the palette or the most distant ones using colorRampPalette.

All four functions can be expanded further, for example with facet_grid or facet_wrap.

Value

This function returns a ggplot2 object of class "gg" and "ggplot".

Examples

```
#3d version for 1-way data with blocking
plot_3d_scatterbox(data = data_1w_death,
xcol = Genotype, ycol = Death, shapes = Experiment)
#compare above graph to
plot_scatterbox(data = data_1w_death, xcol = Genotype, ycol = Death)

#4d version for 2-way data with blocking
plot_4d_scatterbox(data = data_2w_Tdeath,
xcol = Genotype,
ycol = PI,
boxes = Time,
shapes = Experiment)

plot_4d_scatterbar(data = data_2w_Festing,
xcol = Strain,
ycol = GST,
bars = Treatment,
shapes = Block)
```

---

plot_4d_scatterbox *Plot a dot plot with matched shapes on a box plot using four variables.*

Description

The functions `plot_3d_scatterbar`, `plot_3d_scatterbox`, `plot_4d_scatterbar` and `plot_4d_scatterbox` are useful for plotting one-way or two-way ANOVA designs with randomised blocks or repeated measures. The blocks or subjects can be mapped to the `shapes` argument in both functions (up to 25 levels can be mapped to `shapes`; there will be an error if this number is exceeded). The 3d versions use the categorical variable (`xcol`) for grouping (e.g. one-way ANOVA designs), and 4d versions take an additional grouping variable (e.g. two-way ANOVA designs) that is passed to either `boxes` or `bars` argument.

Usage

```
plot_4d_scatterbox(
  data,
  xcol,
  ycol,
  boxes,
  shapes,
  symsize = 2.5,
  symthick = 1,
  jitter = 0.2,
  fontsize = 20,
  b_alpha = 1,
  s_alpha = 1,
  ColSeq = TRUE,
  ColPal = c("okabe_ito", "all_grafify", "bright", "contrast", "dark", "fishy",
    "kelly", "light", "muted", "pale", "r4", "safe", "vibrant"),
  ColRev = FALSE,
  TextXAngle = 0,
  ...
)
```

Arguments

`data`	a data table, e.g. data.frame or tibble.
`xcol`	name of the column with the categorical factor to plot on X axis. If column is numeric, enter as `factor(col)`.
`ycol`	name of the column to plot on quantitative variable on the Y axis.
`boxes`	name of the column containing grouping within the factor plotted on X axis. Can be categorical or numeric X. If your table has numeric X and you want to plot as factor, enter `xcol = factor(name of colum)`.
`shapes`	name of the column that contains matched observations, e.g. subject IDs, experiment number etc.
`symsize`	size of symbols, default set to 3.
`symthick`	size of outline of symbol lines (`stroke = 1.0`), default set to 1.0.
`jitter`	extent of jitter (scatter) of symbols, default is 0.2. Increase to reduce symbol overlap, set to 0 for aligned symbols.

`fontsize`	parameter of `base_size` of fonts in `theme_classic`, default set to size 20.
`b_alpha`	fractional opacity of boxes, default set to 1 (i.e. maximum opacity & zero transparency).
`s_alpha`	fractional opacity of symbols, default set to 1 (i.e. maximum opacity & zero transparency).
`ColSeq`	logical TRUE or FALSE. Default TRUE for sequential colours from chosen palette. Set to FALSE for distant colours, which will be applied using `scale_fill_grafify2`.
`ColPal`	grafify colour palette to apply, default "okabe_ito"; see `graf_palettes` for available palettes.
`ColRev`	whether to reverse order of colour within the selected palette, default F (FALSE); can be set to T (TRUE).
`TextXAngle`	orientation of text on X-axis; default 0 degrees. Change to 45 or 90 to remove overlapping text.
`...`	any additional arguments to pass to `ggplot2`geom_boxplot.

Details

These functions rely on `ggplot` with `geom_point` and `geom_bar` (through `stat_summary`) or `geom_boxplot` geometries.

Variables other than the quantitative variable (`ycol`) will be automatically converted to categorical variables even if they are numeric in the data table.

Shapes are always plotted in black colour, and their opacity can be changed with the `s_alpha` argument and overlap can be reduced with the `jitter` argument. Other arguments are similar to other plot functions as briefly explained below.

Bars depict means using `stat_summary` with `geom = "bar", fun = "mean"` , and bar width is set to 0.7 (cannot be changed). Error bar width can be changed with the `ewid` argument.

Boxplot geometry uses `geom_boxplot` with `position = position_dodge(width = 0.9), width = 0.6`. The thick line within the boxplot depicts the median, the box the IQR (interquantile range) and the whiskers show 1.5*IQR.

In 4d versions, the two grouping variables (i.e. `xcol` and either `boxes` or `bars`) are passed to ggplot aesthetics through `group = interaction{ xcol, shapes}`.

Colours can be changed using `ColPal`, `ColRev` or `ColSeq` arguments. `ColPal` can be one of the following: "okabe_ito", "dark", "light", "bright", "pale", "vibrant, "muted" or "contrast". `ColRev` (logical TRUE/FALSE) decides whether colours are chosen from first-to-last or last-to-first from within the chosen palette. `ColSeq` (logical TRUE/FALSE) decides whether colours are picked by respecting the order in the palette or the most distant ones using `colorRampPalette`.

All four functions can be expanded further, for example with `facet_grid` or `facet_wrap`.

Value

This function returns a `ggplot2` object of class "gg" and "ggplot".

Examples

```
#3d version for 1-way data with blocking
plot_3d_scatterbox(data = data_1w_death,
xcol = Genotype, ycol = Death, shapes = Experiment)
#compare above graph to
plot_scatterbox(data = data_1w_death, xcol = Genotype, ycol = Death)

#4d version for 2-way data with blocking
plot_4d_scatterbox(data = data_2w_Tdeath,
xcol = Genotype,
ycol = PI,
boxes = Time,
shapes = Experiment)

plot_4d_scatterbar(data = data_2w_Festing,
xcol = Strain,
ycol = GST,
bars = Treatment,
shapes = Block)
```

`plot_4d_scatterviolin` *Plot a dot plot with matched shapes on a violin & box plot using four variables.*

Description

The functions `plot_3d_scatterbar`, `plot_3d_scatterbox`, `plot_3d_scatterviolin`, `plot_4d_scatterbar`, `plot_4d_scatterbox` and `plot_4d_scatterviolin` are useful for plotting one-way or two-way ANOVA designs with randomised blocks or repeated measures. The blocks or subjects can be mapped to the `shapes` argument in both functions (up to 25 levels can be mapped to `shapes`; there will be an error if this number is exceeded). The 3d versions use the categorical variable (`xcol`) for grouping (e.g. one-way ANOVA designs), and 4d versions take an additional grouping variable (e.g. two-way ANOVA designs) that is passed to either `boxes` or `bars` argument.

Usage

```
plot_4d_scatterviolin(
  data,
  xcol,
  ycol,
  boxes,
  shapes,
  symsize = 2.5,
  s_alpha = 1,
  symthick = 1,
  jitter = 0.2,
  v_alpha = 1,
  b_alpha = 0,
  bvthick = 1,
  bwid = 0.2,
  ColSeq = TRUE,
  ColPal = c("okabe_ito", "all_grafify", "bright", "contrast", "dark", "fishy",
    "kelly", "light", "muted", "pale", "r4", "safe", "vibrant"),
  ColRev = FALSE,
  TextXAngle = 0,
  scale = "width",
  trim = TRUE,
  fontsize = 20,
  ...
)
```

Arguments

`data`	a data table, e.g. data.frame or tibble.
`xcol`	name of the column with the categorical factor to plot on X axis. If column is numeric, enter as `factor(col)`.
`ycol`	name of the column to plot on quantitative variable on the Y axis.
`boxes`	name of the column containing grouping within the factor plotted on X axis. Can be categorical or numeric X. If your table has numeric X and you want to plot as factor, enter `xcol = factor(name of colum)`.
`shapes`	name of the column that contains matched observations, e.g. subject IDs, experiment number etc.
`symsize`	size of symbols, default set to 3.
`s_alpha`	fractional opacity of symbols, default set to 1 (i.e. maximum opacity & zero transparency). Set `s_alpha = 0` to not show scatter plot.
`symthick`	size of outline of symbol lines (`stroke = 1.0`), default set to 1.0.
`jitter`	extent of jitter (scatter) of symbols, default is 0 (i.e. aligned symbols). To reduce symbol overlap, try 0.1-0.3 or higher.
`v_alpha`	fractional opacity of violins, default set to 1 (i.e. maximum opacity & zero transparency).
`b_alpha`	fractional opacity of boxplots. Default is set to 0, which results in white boxes inside violins. Change to any value >0 up to 1 for different levels of transparency.
`bvthick`	thickness of both violin and boxplot lines; default 1.
`bwid`	width of boxplots; default 0.2
`ColSeq`	logical TRUE or FALSE. Default TRUE for sequential colours from chosen palette. Set to FALSE for distant colours, which will be applied using `scale_fill_grafify2`.
`ColPal`	grafify colour palette to apply, default "okabe_ito"; see `graf_palettes` for available palettes.
`ColRev`	whether to reverse order of colour within the selected palette, default F (FALSE); can be set to T (TRUE).

`TextXAngle`	orientation of text on X-axis; default 0 degrees. Change to 45 or 90 to remove overlapping text.
`scale`	set to "area" by default, can be changed to "count" or "width".
`trim`	set whether tips of violin plot should be trimmed at high/low data. Default `trim = T`, can be changed to F.
`fontsize`	parameter of `base_size` of fonts in `theme_classic`, default set to size 20.
`...`	any additional arguments to pass to `ggplot2`geom_boxplot or `ggplot2`geom_violin.

Details

These functions rely on `ggplot` with `geom_point` and `geom_bar` (through `stat_summary`), or `geom_violin` and `geom_boxplot` geometries.

Variables other than the quantitative variable (`ycol`) will be automatically converted to categorical variables even if they are numeric in the data table.

Shapes are always plotted in black colour, and their opacity can be changed with the `s_alpha` argument and overlap can be reduced with the `jitter` argument. Other arguments are similar to other plot functions as briefly explained below.

Bars depict means using `stat_summary` with `geom = "bar", fun = "mean"` , and bar width is set to 0.7 (cannot be changed). Error bar width can be changed with the `ewid` argument.

Boxplot geometry uses `geom_boxplot` with `position = position_dodge(width = 0.9), width = 0.6`. The thick line within the boxplot depicts the median, the box the IQR (interquantile range) and the whiskers show 1.5*IQR.

In 4d versions, the two grouping variables (i.e. `xcol` and either `boxes` or `bars`) are passed to ggplot aesthetics through `group = interaction{ xcol, shapes}`.

Colours can be changed using `ColPal`, `ColRev` or `ColSeq` arguments. `ColPal` can be one of the following: "okabe_ito", "dark", "light", "bright", "pale", "vibrant, "muted" or "contrast". `ColRev` (logical TRUE/FALSE) decides whether colours are chosen from first-to-last or last-to-first from within the chosen palette. `ColSeq` (logical TRUE/FALSE) decides whether colours are picked by respecting the order in the palette or the most distant ones using `colorRampPalette`.

All four functions can be expanded further, for example with `facet_grid` or `facet_wrap`.

Value

This function returns a `ggplot2` object of class "gg" and "ggplot".

Examples

```
#3d version for 1-way data with blocking
plot_3d_scatterviolin(data = data_1w_death,
xcol = Genotype, ycol = Death, shapes = Experiment)
#compare above graph to
plot_scatterviolin(data = data_1w_death,
xcol = Genotype, ycol = Death)

#4d version for 2-way data with blocking
plot_4d_scatterviolin(data = data_2w_Tdeath,
xcol = Genotype,
ycol = PI,
boxes = Time,
shapes = Experiment)
```

---

plot_bar_sd   *Plot a bar graph indicating mean with error bars (SD) using two variables.*

Description

This function takes a data table, categorical X and numeric Y variables, and plots bars showing the mean with SD error bars. The X variable is mapped to the `fill` aesthetic of bars. The related `plot_bar_sd_sc` plots bars with a single colour.

Usage

```
plot_bar_sd(
  data,
  xcol,
  ycol,
  b_alpha = 1,
  bwid = 0.7,
  bthick = 1,
  ewid = 0.3,
  ColPal = c("okabe_ito", "all_grafify", "bright", "contrast", "dark", "fishy",
    "kelly", "light", "muted", "pale", "r4", "safe", "vibrant"),
  ColSeq = TRUE,
  ColRev = FALSE,
  SingleColour = "NULL",
  TextXAngle = 0,
  fontsize = 20,
  ...
)
```

Arguments

`data`	a data table object, e.g. a data.frame or tibble.
`xcol`	name of the column to plot on X axis. This should be a categorical variable.
`ycol`	name of the column to plot on the Y axis. This should be a quantitative variable.
`b_alpha`	fractional opacity of bars, default set to 1 (i.e. maximum opacity & zero transparency).
`bwid`	width of bars, default 0.7
`bthick`	thickness of bar borders; default 1.
`ewid`	width of error bars, default 0.3

`ColPal`	grafify colour palette to apply, default "okabe_ito"; see `graf_palettes` for available palettes.
`ColSeq`	logical TRUE or FALSE. Default TRUE for sequential colours from chosen palette. Set to FALSE for distant colours, which will be applied using `scale_fill_grafify2`.
`ColRev`	whether to reverse order of colour within the selected palette, default F (FALSE); can be set to T (TRUE).
`SingleColour`	a colour hexcode (starting with #), a number between 1-154, or names of colours from `grafify` colour palettes to fill along X-axis aesthetic. Accepts any colour other than "black"; use `grey_lin11`, which is almost black.
`TextXAngle`	orientation of text on X-axis; default 0 degrees. Change to 45 or 90 to remove overlapping text.
`fontsize`	parameter of `base_size` of fonts in `theme_classic`, default set to size 20.
`...`	any additional arguments to pass to `stat_summary`.

Details

The function uses `stat_summary` with `geom = "bar"`. Standard deviation (SD) is plotted through `stat_summary` calculated using `mean_sdl` from the `ggplot2` package (get help with `?mean_sdl`), and 1x SD is plotted (`fun.arg = list(mult = 1)`.

Colours can be changed using `ColPal`, `ColRev` or `ColSeq` arguments. Colours available can be seen quickly with `plot_grafify_palette`. `ColPal` can be one of the following: "okabe_ito", "dark", "light", "bright", "pale", "vibrant, "muted" or "contrast". `ColRev` (logical TRUE/FALSE) decides whether colours are chosen from first-to-last or last-to-first from within the chosen palette. `ColSeq` decides whether colours are picked by respecting the order in the palette or the most distant ones using `colorRampPalette`.

If there are many groups along the X axis and you prefer a single colour for the graph,use the `SingleColour` argument.

You are instead encouraged to show all data using the following functions: `plot_scatterbar_sd`, `plot_scatterbox`, `plot_dotbox`, `plot_dotbar_sd`, `plot_scatterviolin` or `plot_dotviolin`.

Value

This function returns a `ggplot2` object of class "gg" and "ggplot".

Examples

```
#Basic usage
plot_bar_sd(data = data_doubling_time,
xcol = Student, ycol = Doubling_time)

#apply distant colours in the default palette
plot_bar_sd(data = data_doubling_time,
xcol = Student, ycol = Doubling_time,
ColSeq = FALSE)

#single colour along X axis aesthetic
```

```
plot_bar_sd(data = data_doubling_time,
xcol = Student, ycol = Doubling_time,
SingleColour = "pale_cyan")
```

plot_befafter_colors *Plot a before-after plot with lines joining colour-matched symbols.*

Description

The plot_befafter_colours, plot_befafter_colors and plot_befafter_shapes are for plotting matched data joined by lines. These functions take X and Y variables along with a data column with matching information (e.g. matched subjects or experiments etc.) and plot symbols matched by colour or shape.

Usage

```
plot_befafter_colors(
  data,
  xcol,
  ycol,
  match,
  symsize = 3,
  symthick = 1,
  s_alpha = 1,
  ColPal = c("okabe_ito", "all_grafify", "bright", "contrast", "dark", "fishy",
    "kelly", "light", "muted", "pale", "r4", "safe", "vibrant"),
  ColSeq = TRUE,
  ColRev = FALSE,
  SingleColour = "NULL",
  TextXAngle = 0,
  fontsize = 20,
  groups,
  ...
)
```

Arguments

data	a data table object, e.g. data.frame or tibble.
xcol	name of the column containing the categorical variable to be plotted on the X axis.
ycol	name of the column containing the quantitative Y values.
match	name of the column with the grouping variable to pass on to geom_line.
symsize	size of symbols, default set to 3.
symthick	thickness of symbol border, default set to 1.

`s_alpha`	fractional opacity of symbols, default set to 1 (i.e. maximum opacity & zero transparency).
`ColPal`	grafify colour palette to apply, default "okabe_ito"; see `graf_palettes` for available palettes.
`ColSeq`	logical TRUE or FALSE. Default TRUE for sequential colours from chosen palette. Set to FALSE for distant colours, which will be applied using `scale_fill_grafify2`.
`ColRev`	whether to reverse order of colour within the selected palette, default F (FALSE); can be set to T (TRUE).
`SingleColour`	a colour hexcode (starting with #), a number between 1-154, or names of colours from `grafify` colour palettes to fill along X-axis aesthetic. Accepts any colour other than "black"; use `grey_lin11`, which is almost black.
`TextXAngle`	orientation of text on X-axis; default 0 degrees. Change to 45 or 90 to remove overlapping text.
`fontsize`	parameter of `base_size` of fonts in `theme_classic`, default set to size 20.
`groups`	old argument name for `match`; retained for backward compatibility.
`...`	any additional arguments to pass to `ggplot2`geom_line or `ggplot2`geom_point.

Details

Note that only 25 shapes are available, and there will be errors with `plot_befafter_shapes` when there are fewer than 25 matched observations; instead use `plot_befafter_colours` instead.

Colours can be changed using `ColPal`, `ColRev` or `ColSeq` arguments. `ColPal` can be one of the following: "okabe_ito", "dark", "light", "bright", "pale", "vibrant, "muted" or "contrast". `ColRev` (logical TRUE/FALSE) decides whether colours are chosen from first-to-last or last-to-first from within the chosen palette. `ColSeq` decides whether colours are picked by respecting the order in the palette or the most distant ones using `colorRampPalette`.

To plot a graph with a single colour along the X axis variable, use the `SingleColour` argument.

More complex designs can also be plotted when used with `facet_wrap` or `facet_grid`.

Value

This function returns a `ggplot2` object of class "gg" and "ggplot".

Examples

```
#plot without legends if necessary
plot_befafter_colors(data = data_t_pdiff,
xcol = Condition, ycol = Mass,
match = Subject, s_alpha = .9, ColSeq = FALSE)+
guides(fill = "none",
colour = "none") #remove guides
#2way ANOVA design with randomised blocks
plot_befafter_colors(data = data_2w_Tdeath,
xcol = Genotype, ycol = PI,
match = Experiment) + facet_wrap("Time")
```

plot_befafter_colours *Plot a before-after plot with lines joining colour-matched symbols.*

Description

The plot_befafter_colours, plot_befafter_colors and plot_befafter_shapes are for plotting matched data joined by lines. These functions take X and Y variables along with a data column with matching information (e.g. matched subjects or experiments etc.) and plot symbols matched by colour or shape.

Usage

```
plot_befafter_colours(
  data,
  xcol,
  ycol,
  match,
  symsize = 3,
  symthick = 1,
  s_alpha = 1,
  ColPal = c("okabe_ito", "all_grafify", "bright", "contrast", "dark", "fishy",
    "kelly", "light", "muted", "pale", "r4", "safe", "vibrant"),
  ColSeq = TRUE,
  ColRev = FALSE,
  SingleColour = "NULL",
  TextXAngle = 0,
  fontsize = 20,
  groups,
  ...
)
```

Arguments

`data`	a data table object, e.g. data.frame or tibble.
`xcol`	name of the column containing the categorical variable to be plotted on the X axis.
`ycol`	name of the column containing the quantitative Y values.
`match`	name of the column with the matching variable to pass on to `geom_line`.
`symsize`	size of symbols, default set to 3.
`symthick`	thickness of symbol border, default set to 1.
`s_alpha`	fractional opacity of symbols, default set to 1 (i.e. maximum opacity & zero transparency).
`ColPal`	grafify colour palette to apply, default "okabe_ito"; see graf_palettes for available palettes.

`ColSeq`	logical TRUE or FALSE. Default TRUE for sequential colours from chosen palette. Set to FALSE for distant colours, which will be applied using `scale_fill_grafify2`.
`ColRev`	whether to reverse order of colour within the selected palette, default F (FALSE); can be set to T (TRUE).
`SingleColour`	a colour hexcode (starting with #), a number between 1-154, or names of colours from `grafify` colour palettes to fill along X-axis aesthetic. Accepts any colour other than "black"; use `grey_lin11`, which is almost black.
`TextXAngle`	orientation of text on X-axis; default 0 degrees. Change to 45 or 90 to remove overlapping text.
`fontsize`	parameter of `base_size` of fonts in `theme_classic`, default set to size 20.
`groups`	old argument name for `match`; retained for backward compatibility.
`...`	any additional arguments to pass to `ggplot2`geom_line or `ggplot2`geom_point.

Details

Note that only 25 shapes are available, and there will be errors with `plot_befafter_shapes` when there are fewer than 25 matched observations; instead use `plot_befafter_colours` instead.

Colours can be changed using `ColPal`, `ColRev` or `ColSeq` arguments. `ColPal` can be one of the following: "okabe_ito", "dark", "light", "bright", "pale", "vibrant, "muted" or "contrast". `ColRev` (logical TRUE/FALSE) decides whether colours are chosen from first-to-last or last-to-first from within the chosen palette. `ColSeq` decides whether colours are picked by respecting the order in the palette or the most distant ones using `colorRampPalette`.

To plot a graph with a single colour along the X axis variable, use the `SingleColour` argument.

More complex designs can also be plotted when used with `facet_wrap` or `facet_grid`.

Value

This function returns a `ggplot2` object of class "gg" and "ggplot".

Examples

```
#plot without legends if necessary
plot_befafter_colors(data = data_t_pdiff,
xcol = Condition, ycol = Mass,
match = Subject, s_alpha = .9, ColSeq = FALSE)+
guides(fill = "none",
colour = "none") #remove guides
#2way ANOVA design with randomised blocks
plot_befafter_colors(data = data_2w_Tdeath,
xcol = Genotype, ycol = PI,
match = Experiment) + facet_wrap("Time")
```

plot_befafter_shapes *Plot a before-after plot with lines joining shape-matched symbols.*

Description

The plot_befafter_colours, plot_befafter_colors and plot_befafter_shapes are for plotting matched data joined by lines. These functions take X and Y variables along with a data column with matching information (e.g. matched subjects or experiments etc.) and plot symbols matched by colour or shape.

Usage

```
plot_befafter_shapes(
  data,
  xcol,
  ycol,
  match,
  symsize = 3,
  symthick = 1,
  s_alpha = 0.8,
  ColPal = c("okabe_ito", "all_grafify", "bright", "contrast", "dark", "fishy",
    "kelly", "light", "muted", "pale", "r4", "safe", "vibrant"),
  ColSeq = TRUE,
  ColRev = FALSE,
  SingleColour = "NULL",
  TextXAngle = 0,
  fontsize = 20,
  groups,
  ...
)
```

Arguments

`data`	a data table object, e.g. data.frame or tibble.
`xcol`	name of the column containing the categorical variable to be plotted on the X axis.
`ycol`	name of the column containing the quantitative Y values.
`match`	name of the column with the matching variable to pass on to `geom_line`.
`symsize`	size of symbols, default set to 3.
`symthick`	size of outline of symbol lines (`stroke` = 1), default set to 1.
`s_alpha`	fractional opacity of symbols, default set to 0.8 (i.e. 80% opacity).
`ColPal`	grafify colour palette to apply, default "okabe_ito"; see graf_palettes for available palettes.
`ColSeq`	logical TRUE or FALSE. Default TRUE for sequential colours from chosen palette. Set to FALSE for distant colours, which will be applied using `scale_fill_grafify2`.

`ColRev`	whether to reverse order of colour within the selected palette, default F (FALSE); can be set to T (TRUE).
`SingleColour`	a colour hexcode (starting with #), a number between 1-154, or names of colours from `grafify` colour palettes to fill along X-axis aesthetic. Accepts any colour other than "black"; use `grey_lin11`, which is almost black.
`TextXAngle`	orientation of text on X-axis; default 0 degrees. Change to 45 or 90 to remove overlapping text.
`fontsize`	parameter of `base_size` of fonts in `theme_classic`, default set to size 20.
`groups`	old argument name for `match`; retained for backward compatibility.
`...`	any additional arguments to pass to `ggplot2`geom_line or `ggplot2`geom_point.

Details

Note that only 25 shapes are available, and there will be errors with `plot_befafter_shapes` when there are fewer than 25 matched observations; instead use `plot_befafter_colours` instead.

Colours can be changed using `ColPal`, `ColRev` or `ColSeq` arguments. `ColPal` can be one of the following: "okabe_ito", "dark", "light", "bright", "pale", "vibrant, "muted" or "contrast". `ColRev` (logical TRUE/FALSE) decides whether colours are chosen from first-to-last or last-to-first from within the chosen palette. `ColSeq` decides whether colours are picked by respecting the order in the palette or the most distant ones using `colorRampPalette`.

To plot a graph with a single colour along the X axis variable, use the `SingleColour` argument.

More complex designs can also be plotted when used with `facet_wrap` or `facet_grid`.

Value

This function returns a `ggplot2` object of class "gg" and "ggplot".

Examples

```
#plot without legends if necessary
plot_befafter_colors(data = data_t_pdiff,
xcol = Condition, ycol = Mass,
match = Subject, s_alpha = .9, ColSeq = FALSE)+
guides(fill = "none",
colour = "none") #remove guides
#2way ANOVA design with randomised blocks
plot_befafter_colors(data = data_2w_Tdeath,
xcol = Genotype, ycol = PI,
match = Experiment) + facet_wrap("Time")
```

plot_density *Plot density distribution of data.*

Description

This function takes a data table, `ycol` of quantitative variable and a categorical grouping variable (`group`), if available, and plots a density graph using `geom_density`).

Usage

```
plot_density(
  data,
  ycol,
  group,
  linethick = 1,
  c_alpha = 0.2,
  ColPal = c("okabe_ito", "all_grafify", "bright", "contrast", "dark", "fishy",
    "kelly", "light", "muted", "pale", "r4", "safe", "vibrant"),
  ColSeq = TRUE,
  ColRev = FALSE,
  TextXAngle = 0,
  fontsize = 20,
  Group,
  alpha,
  ...
)
```

Arguments

`data`	a data table e.g. data.frame or tibble.
`ycol`	name of the column containing the quantitative variable whose density distribution is to be plotted.
`group`	name of the column containing a categorical grouping variable
`linethick`	thickness of symbol border, default set to 1.
`c_alpha`	fractional opacity of filled colours under the curve, default set to 0.2 (i.e. 20% opacity).
`ColPal`	grafify colour palette to apply, default "okabe_ito"; see `graf_palettes` for available palettes.
`ColSeq`	logical TRUE or FALSE. Default TRUE for sequential colours from chosen palette. Set to FALSE for distant colours, which will be applied using `scale_fill_grafify2`.
`ColRev`	whether to reverse order of colour within the selected palette, default F (FALSE); can be set to T (TRUE).
`TextXAngle`	orientation of text on X-axis; default 0 degrees. Change to 45 or 90 to remove overlapping text.

`fontsize`	parameter of `base_size` of fonts in `theme_classic`, default set to size 20.
`Group`	deprecated old argument for `group`; retained for backward compatibility.
`alpha`	deprecated old argument for `c_alpha`; retained for backward compatibility.
`...`	any additional arguments to pass to `ggplot2`geom_density.

Details

Note that the function requires the quantitative Y variable first, and groups them based on an X variable. The group variable is mapped to the `fill` and `colour` aesthetics in `geom_density`. Colours can be changed using `ColPal`, `ColRev` or `ColSeq` arguments. Colours available can be seen quickly with `plot_grafify_palette`. `ColPal` can be one of the following: "okabe_ito", "dark", "light", "bright", "pale", "vibrant, "muted" or "contrast". `ColRev` (logical TRUE/FALSE) decides whether colours are chosen from first-to-last or last-to-first from within the chosen palette. `ColSeq` decides whether colours are picked by respecting the order in the palette or the most distant ones using `colorRampPalette`.

Value

This function returns a `ggplot2` object of class "gg" and "ggplot".

Examples

```
plot_density(data = data_t_pratio,
ycol = log(Cytokine), group = Genotype)

#with faceting
plot_density(data = data_cholesterol,
ycol = Cholesterol, group = Treatment,
fontsize = 10)+facet_wrap("Treatment")
```

`plot_dotbar_sd` *Plot a dotplot on a bar graph with SD error bars with two variables.*

Description

This function takes a data table, X and Y variables, and plots a graph with a dotplot and bars using `stat_summary` with `geom = "bar"`, and `geom_dotplot` geometries. Standard deviation (SD) is plotted through `stat_summary` calculated using `mean_sdl` from the `ggplot2` package (get help with ?`mean_sdl`), and 1x SD is plotted (`fun.arg = list(mult = 1)`.

Usage

```
plot_dotbar_sd(
  data,
  xcol,
  ycol,
  dotsize = 1.5,
  dotthick = 1,
  bwid = 0.7,
  ewid = 0.2,
  b_alpha = 1,
  d_alpha = 1,
  ColPal = c("okabe_ito", "all_grafify", "bright", "contrast", "dark", "fishy",
    "kelly", "light", "muted", "pale", "r4", "safe", "vibrant"),
  ColRev = FALSE,
  ColSeq = TRUE,
  SingleColour = "NULL",
  TextXAngle = 0,
  fontsize = 20,
  ...
)
```

Arguments

`data`	a data table object, e.g. data.frame or tibble.
`xcol`	name of the column to plot on X axis. This should be a categorical variable.
`ycol`	name of the column to plot on quantitative Y axis. This should be a quantitative variable.
`dotsize`	size of dots relative to binwidth used by `geom_dotplot`. Default set to 1.5, increase/decrease as needed.
`dotthick`	thickness of dot border (`stroke` parameter of `geom_dotplot`), default set to 1.
`bwid`	width of bars, default set to 0.7
`ewid`	width of error bars, default set to 0.2.
`b_alpha`	fractional opacity of bars, default set to 1 (i.e. maximum opacity & zero transparency).
`d_alpha`	fractional opacity of dots, default set to 1 (i.e. maximum opacity & zero transparency).
`ColPal`	grafify colour palette to apply, default "okabe_ito"; see `graf_palettes` for available palettes.
`ColRev`	whether to reverse order of colour within the selected palette, default F (FALSE); can be set to T (TRUE).
`ColSeq`	logical TRUE or FALSE. Default TRUE for sequential colours from chosen palette. Set to FALSE for distant colours, which will be applied using `scale_fill_grafify2`.
`SingleColour`	a colour hexcode (starting with #), a number between 1-154, or names of colours from `grafify` colour palettes to fill along X-axis aesthetic. Accepts any colour other than "black"; use `grey_lin11`, which is almost black.
`TextXAngle`	orientation of text on X-axis; default 0 degrees. Change to 45 or 90 to remove overlapping text.
`fontsize`	parameter of `base_size` of fonts in `theme_classic`, default set to size 20.
`...`	any additional arguments to pass to `ggplot2`geom_dotplot.

Details

The X variable is mapped to the `fill` aesthetic in both bar and dotplot.

Colours can be changed using `ColPal`, `ColRev` or `ColSeq` arguments. Colours available can be seen quickly with `plot_grafify_palette`. `ColPal` can be one of the following: "okabe_ito", "dark", "light", "bright", "pale", "vibrant, "muted" or "contrast". `ColRev` (logical TRUE/FALSE) decides whether colours are chosen from first-to-last or last-to-first from within the chosen palette. `ColSeq` decides whether colours are picked by respecting the order in the palette or the most distant ones using `colorRampPalette`.

If there are many groups along the X axis and you prefer a single colour for the graph,use the `SingleColour` argument.

The size of dots can be adjusted using the parameter, which is `dotsize = 1` by default.

This function is related to `plot_dotbar_sd` and `plot_dotviolin`.

Value

This function returns a `ggplot2` object of class "gg" and "ggplot".

Examples

```
plot_dotbar_sd(data = data_cholesterol,
xcol = Treatment,
ycol = Cholesterol)

plot_dotbar_sd(data = data_1w_death,
xcol = Genotype, ycol = Death,
ColPal = "pale", ColSeq = FALSE, ColRev = TRUE)

#single colour along X
plot_dotbar_sd(data = data_1w_death,
xcol = Genotype, ycol = Death,
SingleColour = "light_orange")
```

`plot_dotbox` *Plot a dotplot on a boxplot with two variables.*

Description

This function takes a data table, X and Y variables, and plots a graph with a dotplot and boxplot using `geom_boxplot` and `geom_dotplot` geometries. Note that `geom_boxplot` option for outliers is set to `outlier.alpha = 0`.

Usage

```
plot_dotbox(
  data,
  xcol,
  ycol,
  dotsize = 1.5,
  dotthick = 1,
  b_alpha = 1,
  d_alpha = 1,
  ColPal = c("okabe_ito", "all_grafify", "bright", "contrast", "dark", "fishy",
    "kelly", "light", "muted", "pale", "r4", "safe", "vibrant"),
  ColRev = FALSE,
  ColSeq = TRUE,
  SingleColour = "NULL",
  TextXAngle = 0,
  fontsize = 20,
  ...
)
```

Arguments

`data`	a data table object, e.g. data.frame or tibble.
`xcol`	name of the column to plot on X axis. This should be a categorical variable.
`ycol`	name of the column to plot on quantitative Y axis. This should be a quantitative variable.
`dotsize`	size of dots relative to binwidth used by `geom_dotplot`. Default set to 1.5, increase/decrease as needed.
`dotthick`	thickness of dot border (`stroke` parameter of `geom_dotplot`), default set to 1.
`b_alpha`	fractional opacity of boxes, default set to 1 (i.e. maximum opacity & zero transparency).
`d_alpha`	fractional opacity of dots, default set to 1 (i.e. maximum opacity & zero transparency).
`ColPal`	grafify colour palette to apply, default "okabe_ito"; see `graf_palettes` for available palettes.
`ColRev`	whether to reverse order of colour within the selected palette, default F (FALSE); can be set to T (TRUE).
`ColSeq`	logical TRUE or FALSE. Default TRUE for sequential colours from chosen palette. Set to FALSE for distant colours, which will be applied using `scale_fill_grafify2`.
`SingleColour`	a colour hexcode (starting with #), a number between 1-154, or names of colours from `grafify` colour palettes to fill along X-axis aesthetic. Accepts any colour other than "black"; use `grey_lin11`, which is almost black.
`TextXAngle`	orientation of text on X-axis; default 0 degrees. Change to 45 or 90 to remove overlapping text.
`fontsize`	parameter of `base_size` of fonts in `theme_classic`, default set to size 20.
`...`	any additional arguments to pass to `ggplot2`geom_boxplot or `ggplot2`geom_dotplot.

Details

The X variable is mapped to the `fill` aesthetic in both boxplot and dotplot. Colours can be changed using `ColPal`, `ColRev` or `ColSeq` arguments. Colours available can be seen quickly with `plot_grafify_palette`. `ColPal` can be one of the following: "okabe_ito", "dark", "light", "bright", "pale", "vibrant, "muted" or "contrast". `ColRev` (logical TRUE/FALSE) decides whether colours are chosen from first-to-last or last-to-first from within the chosen palette. `ColSeq` decides whether colours are picked by respecting the order in the palette or the most distant ones using `colorRampPalette`.

If there are many groups along the X axis and you prefer a single colour for the graph,use the `SingleColour` argument.

The size of dots can be adjusted using the parameter, which is `dotsize = 1` by default.

This function is related to `plot_dotbar_sd` and `plot_dotviolin`.

Value

This function returns a `ggplot2` object of class "gg" and "ggplot".

Examples

```
plot_dotbox(data = data_1w_death,
xcol = Genotype, ycol = Death)

plot_dotbox(data = data_1w_death,
xcol = Genotype, ycol = Death,
ColPal = "vibrant", b_alpha = 0.5)
plot_dotbox(data = data_1w_death,
xcol = Genotype, ycol = Death,
SingleColour = "safe_bluegreen", b_alpha = 0.5)
```

`plot_dotviolin` *Plot a dotplot on a violin plot with two variables.*

Description

This function takes a data table, X and Y variables, and plots a graph with a dotplot, box-whiskers and violinplot using `geom_violin`, `geom_boxplot` `geom_dotplot` geometries.

Usage

```
plot_dotviolin(
  data,
  xcol,
  ycol,
  dotsize = 1.5,
  dotthick = 1,
  bvthick = 1,
  bwid = 0.2,
  trim = TRUE,
  scale = "width",
  b_alpha = 0,
  v_alpha = 1,
  d_alpha = 1,
  ColPal = c("okabe_ito", "all_grafify", "bright", "contrast", "dark", "fishy",
    "kelly", "light", "muted", "pale", "r4", "safe", "vibrant"),
  ColRev = FALSE,
  ColSeq = TRUE,
  SingleColour = "NULL",
  TextXAngle = 0,
  fontsize = 20,
  ...
)
```

Arguments

`data`	a data table object, e.g. data.frame or tibble.
`xcol`	name of the column to plot on X axis. This should be a categorical variable.
`ycol`	name of the column to plot on quantitative Y axis. This should be a quantitative variable.
`dotsize`	size of dots relative to `binwidth` used by `geom_dotplot`. Default set to 1.5, increase/decrease as needed.
`dotthick`	thickness of dot border (`stroke` parameter of `geom_dotplot`), default set to 1.
`bvthick`	thickness of violin and boxplot lines; default 1.
`bwid`	width of boxplots; default 0.2
`trim`	set whether tips of violin plot should be trimmed at high/low data. Default `trim = T`, can be changed to F.
`scale`	set to "area" by default, can be changed to "count" or "width".
`b_alpha`	fractional opacity of boxplots. Default is set to 0, which results in white boxes inside violins. Change to any value >0 up to 1 for different levels of transparency.
`v_alpha`	fractional opacity of violins, default set to 1 (i.e. maximum opacity & zero transparency).
`d_alpha`	fractional opacity of dots, default set to 1 (i.e. maximum opacity & zero transparency).
`ColPal`	grafify colour palette to apply, default "okabe_ito"; see `graf_palettes` for available palettes.
`ColRev`	whether to reverse order of colour within the selected palette, default F (FALSE); can be set to T (TRUE).
`ColSeq`	logical TRUE or FALSE. Default TRUE for sequential colours from chosen palette. Set to FALSE for distant colours, which will be applied using `scale_fill_grafify2`.
`SingleColour`	a colour hexcode (starting with #), a number between 1-154, or names of colours from `grafify` colour palettes to fill along X-axis aesthetic. Accepts any colour other than "black"; use `grey_lin11`, which is almost black.

`TextXAngle`	orientation of text on X-axis; default 0 degrees. Change to 45 or 90 to remove overlapping text.
`fontsize`	parameter of `base_size` of fonts in `theme_classic`, default set to size 20.
`...`	any additional arguments to pass to `ggplot2`geom_boxplot, `ggplot2`geom_dotplot or `ggplot2`geom_violin.

Details

Note that the `geom_violin` options are set as follows: `scale = "width"`. The `trim = T` set by default can be changed when calling the function. The boxplot shows IQR, and whiskers show 1.5*IQR; the median is marked with a thicker horizontal line.

The X variable is mapped to the `fill` aesthetic in both violinplot and dotplot. Colours can be changed using `ColPal`, `ColRev` or `ColSeq` arguments. Colours available can be seen quickly with `plot_grafify_palette`. `ColPal` can be one of the following: "okabe_ito", "dark", "light", "bright", "pale", "vibrant, "muted" or "contrast". `ColRev` (logical TRUE/FALSE) decides whether colours are chosen from first-to-last or last-to-first from within the chosen palette. `ColSeq` (logical TRUE/FALSE) decides whether colours are picked by respecting the order in the palette or the most distant ones using `colorRampPalette`.

If there are many groups along the X axis and you prefer a single colour for the graph,use the `SingleColour` argument.

The size of dots can be adjusted using the parameter, which is `dotsize = 1` by default.

This function is related to `plot_scatterbar_sd`, `plot_dotbar_sd` and `plot_dotviolin`.

Value

This function returns a `ggplot2` object of class "gg" and "ggplot".

Examples

```
#plot with trim = FALSE
plot_dotviolin(data = data_t_pdiff,
xcol = Condition, ycol = Mass,
dotsize = 2, trim = FALSE)

plot_dotviolin(data = data_t_pdiff,
xcol = Condition, ycol = Mass,
trim = FALSE, b_alpha = 0.5,
ColPal = "pale", ColSeq = FALSE)

#single colour along X
plot_dotviolin(data = data_t_pdiff,
xcol = Condition, ycol = Mass,
trim = FALSE, b_alpha = 0.5,
SingleColour = "pale_cyan")
```

`plot_grafify_palette` *See grafify colour palettes*

Description

This simple function allows quick visualisation of colours in grafify palettes and their hex codes. It uses `plot_bar_sd` and some arguments are similar and can be adjusted.

Usage

```
plot_grafify_palette(
  palette = c("okabe_ito", "all_grafify", "bright", "contrast", "dark", "fishy",
    "kelly", "light", "muted", "pale", "r4", "safe", "vibrant"),
  bthick = 0,
  fontsize = 14,
  ...
)
```

Arguments

`palette`	name of grafify palettes: "okabe_ito", "vibrant, "bright", "pale", "muted", "dark", "light", "contrast" or "all_grafify".
`bthick`	thickness of bars; passed on `plot_bar_sd`.
`fontsize`	font size.
`...`	any additional parameters to pass to `plot_bar_sd`

Value

This function returns a `ggplot2` object of class "gg" and "ggplot".

Examples

```
plot_grafify_palette("pale")
plot_grafify_palette("contrast")
```

`plot_histogram` *Plot data distribution as histograms.*

Description

This function takes a data table, a quantitative variable (`ycol`) and a Grouping variable (`group`), if available, and plots a histogram graph using `geom_histogram`).

Usage

```
plot_histogram(
  data,
  ycol,
  group,
  BinSize = 30,
  linethick = 1,
  c_alpha = 0.2,
  ColPal = c("okabe_ito", "all_grafify", "bright", "contrast", "dark", "fishy",
    "kelly", "light", "muted", "pale", "r4", "safe", "vibrant"),
  ColRev = FALSE,
  ColSeq = TRUE,
  TextXAngle = 0,
  fontsize = 20,
  Group,
  alpha,
  ...
)
```

Arguments

`data`	a data table e.g. data.frame or tibble.
`ycol`	name of the column containing the quantitative variable whose histogram distribution is to be plotted.
`group`	name of the column containing a categorical grouping variable.
`BinSize`	bins to use on X-axis, default set to 30.
`linethick`	thickness of symbol border, default set to 1.
`c_alpha`	fractional opacity of colour filled within histograms, default set to 0.2 (i.e. 20% opacity).
`ColPal`	grafify colour palette to apply, default "okabe_ito"; see `graf_palettes` for available palettes.
`ColRev`	whether to reverse order of colour within the selected palette, default F (FALSE); can be set to T (TRUE).
`ColSeq`	logical TRUE or FALSE. Default TRUE for sequential colours from chosen palette. Set to FALSE for distant colours, which will be applied using `scale_fill_grafify2`.
`TextXAngle`	orientation of text on X-axis; default 0 degrees. Change to 45 or 90 to remove overlapping text.
`fontsize`	parameter of `base_size` of fonts in `theme_classic`, default set to size 20.
`Group`	deprecated old argument for `group`; retained for backward compatibility.
`alpha`	deprecated old argument for `c_alpha`; retained for backward compatibility.
`...`	any additional arguments to pass to `ggplot2`geom_histogram.

Details

Note that the function requires the quantitative Y variable first, and groups them based on an X variable. The group variable is mapped to the `fill` and `colour` aesthetics in `geom_histogram`.

ColPal & ColRev options are applied to both `fill` and `colour` scales. Colours available can be seen quickly with `plot_grafify_palette`. Colours can be changed using `ColPal`, `ColRev` or `ColSeq` arguments. `ColPal` can be one of the following: "okabe_ito", "dark", "light", "bright", "pale", "vibrant, "muted" or "contrast". `ColRev` (logical TRUE/FALSE) decides whether colours are chosen from first-to-last or last-to-first from within the chosen palette. `ColSeq` decides whether colours are picked by respecting the order in the palette or the most distant ones using `colorRampPalette`.

Value

This function returns a `ggplot2` object of class "gg" and "ggplot".

Examples

```
#Basic usage
plot_histogram(data = data_t_pratio,
ycol = Cytokine, group = Genotype,
BinSize = 10)
#with log transformation
plot_histogram(data = data_t_pratio,
ycol = log(Cytokine), group = Genotype,
BinSize = 10)
```

`plot_point_sd` *Plot a point as mean with SD error bars using two variables.*

Description

This function takes a data table, categorical X and numeric Y variables, and plots a point showing the mean with SD error bars. The X variable is mapped to the `fill` aesthetic of symbols. The related `plot_point_sd_sc` plots bars with a single colour.

Usage

```
plot_point_sd(
  data,
  xcol,
  ycol,
  s_alpha = 1,
  symsize = 3.5,
  symthick = 1,
  ewid = 0.2,
  ColPal = c("okabe_ito", "all_grafify", "bright", "contrast", "dark", "fishy",
    "kelly", "light", "muted", "pale", "r4", "safe", "vibrant"),
  ColSeq = TRUE,
  ColRev = FALSE,
  SingleColour = "NULL",
  TextXAngle = 0,
  fontsize = 20,
  ...
)
```

Arguments

`data`	a data table object, e.g. data.frame or tibble.
`xcol`	name of the column with a categorical X variable.
`ycol`	name of the column with quantitative Y variable.
`s_alpha`	fractional opacity of symbols, default set to 1 (i.e. maximum opacity & zero transparency).
`symsize`	size of point symbols, default set to 3.5.
`symthick`	thickness of symbol border, default set to 1.
`ewid`	width of error bars, default set to 0.2.
`ColPal`	grafify colour palette to apply, default "okabe_ito"; see `graf_palettes` for available palettes.
`ColSeq`	logical TRUE or FALSE. Default TRUE for sequential colours from chosen palette. Set to FALSE for distant colours, which will be applied using `scale_fill_grafify2`.
`ColRev`	whether to reverse order of colour within the selected palette, default F (FALSE); can be set to T (TRUE).
`SingleColour`	a colour hexcode (starting with #), a number between 1-154, or names of colours from `grafify` colour palettes to fill along X-axis aesthetic. Accepts any colour other than "black"; use `grey_lin11`, which is almost black.
`TextXAngle`	orientation of text on X-axis; default 0 degrees. Change to 45 or 90 to remove overlapping text.
`fontsize`	parameter of `base_size` of fonts in `theme_classic`, default set to size 20.
`...`	any additional arguments to pass to `ggplot2`stat_summary.

Details

The function uses `stat_summary` with `geom = "point"` with `size = 3`. Standard deviation (SD) is plotted through `stat_summary` calculated using `mean_sdl` from the `ggplot2` package (get help with ?`mean_sdl`), and 1x SD is plotted (`fun.arg = list(mult = 1)`).

Colours can be changed using `ColPal`, `ColRev` or `ColSeq` arguments. Colours available can be seen quickly with `plot_grafify_palette`. `ColPal` can be one of the following: "okabe_ito", "dark", "light", "bright", "pale", "vibrant, "muted" or "contrast". `ColRev` (logical TRUE/FALSE) decides whether colours are chosen from first-to-last or last-to-first from within the chosen palette. `ColSeq` (logical TRUE/FALSE) decides whether colours are picked by respecting the order in the palette or the most distant ones using `colorRampPalette`.

If there are many groups along the X axis and you prefer a single colour for the graph,use the `SingleColour` argument.

You are instead encouraged to show all data using the following functions: `plot_scatterbar_sd`, `plot_scatterbox`, `plot_dotbox`, `plot_dotbar_sd`, `plot_scatterviolin` or `plot_dotviolin`.

Value

This function returns a ggplot2 object of class "gg" and "ggplot".

Examples

```
#Basic usage
plot_point_sd(data = data_doubling_time,
xcol = Student, ycol = Doubling_time)
```

`plot_qqline` *Plot quantile-quantile (QQ) graphs from data.*

Description

This function takes a data table, a quantitative variable (ycol), and a categorical grouping variable (group), if available, and plots a QQ graph using ggplot2[geom_qq] and ggplot2[geom_qq_line].

Usage

```
plot_qqline(
  data,
  ycol,
  group,
  symsize = 3,
  symthick = 1,
  s_alpha = 1,
  ColPal = c("okabe_ito", "all_grafify", "bright", "contrast", "dark", "fishy",
    "kelly", "light", "muted", "pale", "r4", "safe", "vibrant"),
  ColSeq = TRUE,
  ColRev = FALSE,
  TextXAngle = 0,
  fontsize = 20,
  Group,
  ...
)
```

Arguments

data	a data table e.g. data.frame or tibble.
ycol	name of the column containing the quantitative variable whose distribution is to be plotted.
group	name of the column containing a categorical grouping variable.
symsize	size of symbols, default set to 3.
symthick	thickness of symbol border, default set to 1.

`s_alpha`	fractional opacity of symbols, default set to 1 (i.e. maximum opacity & zero transparency).
`ColPal`	grafify colour palette to apply, default "okabe_ito"; see `graf_palettes` for available palettes.
`ColSeq`	logical TRUE or FALSE. Default TRUE for sequential colours from chosen palette. Set to FALSE for distant colours, which will be applied using `scale_fill_grafify2`.
`ColRev`	whether to reverse order of colour within the selected palette, default F (FALSE); can be set to T (TRUE).
`TextXAngle`	orientation of text on X-axis; default 0 degrees. Change to 45 or 90 to remove overlapping text.
`fontsize`	parameter of `base_size` of fonts in `theme_classic`, default set to size 20.
`Group`	deprecated old argument for `group`; retained for backward compatibility.
`...`	any additional arguments to pass to `ggplot2[geom_qq]` or `ggplot2[geom_qq_line]`.

Details

Note that the function requires the quantitative Y variable first, and can be passed on a grouping variable as `group` if required. The graph plots sample quantiles on Y axis & theoretical quantiles on X axis. The X variable is mapped to the `fill` aesthetic in`stat_qq` and `colour` aesthetic for the `stat_qq_line`.

Colours can be changed using `ColPal`, `ColRev` or `ColSeq` arguments. Colours available can be seen quickly with `plot_grafify_palette`. When only one level is present within `group`, symbols will receive "ok_orange" colour. `ColPal` can be one of the following: "okabe_ito", "dark", "light", "bright", "pale", "vibrant, "muted" or "contrast". `ColRev` (logical TRUE/FALSE) decides whether colours are chosen from first-to-last or last-to-first from within the chosen palette. `ColSeq` decides whether colours are picked by respecting the order in the palette or the most distant ones using `colorRampPalette`.

Value

This function returns a `ggplot2` object of class "gg" and "ggplot".

Examples

```
plot_qqline(data = data_cholesterol,
ycol = Cholesterol, group = Treatment)

#with faceting
plot_qqline(data = data_cholesterol,
ycol = Cholesterol, group = Treatment,
fontsize = 10)+facet_wrap("Treatment")
```

plot_qqmodel — *Plot quantile-quantile (QQ) graphs from residuals of linear models.*

Description

This function takes a linear model (simple or mixed effects) and plots a QQ graph after running `rstudent` from `rstudent` to generate a table of studentized model residuals on an ordinary (`simple_model`), mixed model (`mixed_model` or `mixed_model_slopes`. The graph plots studentized residuals from the model (sample) on Y axis & Theoretical quantiles on X axis.

Usage

```
plot_qqmodel(Model, symsize = 2.5, symthick = 1, s_alpha = 1, fontsize = 20)
```

Arguments

`Model`	name of a saved model generated by `simple_model` or `mixed_model`.
`symsize`	size of symbols, default set to 3.
`symthick`	thickness of symbol border, default set to 1.
`s_alpha`	fractional opacity of symbols, default set to 1 (i.e. maximum opacity & zero transparency).
`fontsize`	parameter of `base_size` of fonts in `theme_classic`, default set to size 20.

Details

The function uses `ggplot2`[geom_qq] and `ggplot2`[geom_qq_line] geometries. Symbols receive "ok_orange" colour by default.

Value

This function returns a `ggplot2` object of class "gg" and "ggplot".

Examples

```
#Basic usage
m1 <- simple_model(data = data_2w_Festing,
Y_value = "GST",
Fixed_Factor = c("Treatment", "Strain"))
plot_qqmodel(m1)
```

`plot_scatterbar_sd` *Plot scatter dots on a bar graph with SD error bars with two variables.*

Description

This function takes a data table, categorical X and numeric Y variables, and plots a graph with a jitterplot or scatterplot and bars showing means with SD error bars. It uses `stat_summary` with `geom = "bar"`, and `geom_point` with `position = position_jitter(width = 0.05)`.

Usage

```
plot_scatterbar_sd(
  data,
  xcol,
  ycol,
  symsize = 2.5,
  symthick = 1,
  bwid = 0.7,
  ewid = 0.3,
  jitter = 0.2,
  b_alpha = 1,
  s_alpha = 1,
  ColPal = c("okabe_ito", "all_grafify", "bright", "contrast", "dark", "fishy",
    "kelly", "light", "muted", "pale", "r4", "safe", "vibrant"),
  ColSeq = TRUE,
  ColRev = FALSE,
  SingleColour = "NULL",
  TextXAngle = 0,
  fontsize = 20
)
```

Arguments

`data`	a data table object, e.g. data.frame or tibble.
`xcol`	name of the column to plot on X axis. This should be a categorical variable.
`ycol`	name of the column to plot on quantitative Y axis. This should be a quantitative variable.
`symsize`	size of point symbols, default set to 2.
`symthick`	thickness of symbol border, default set to 1.
`bwid`	width of bars, default set to 0.7.
`ewid`	width of error bars, default set to 0.3.
`jitter`	extent of jitter (scatter) of symbols, default is 0 (i.e. aligned symbols). To reduce symbol overlap, try 0.1-0.3 or higher.
`b_alpha`	fractional opacity of bars, default set to 1 (i.e. maximum opacity & zero transparency).

`s_alpha`	fractional opacity of symbols, default set to 1 (i.e. maximum opacity & zero transparency).
`ColPal`	grafify colour palette to apply, default "okabe_ito"; see `graf_palettes` for available palettes.
`ColSeq`	logical TRUE or FALSE. Default TRUE for sequential colours from chosen palette. Set to FALSE for distant colours, which will be applied using `scale_fill_grafify2`.
`ColRev`	whether to reverse order of colour within the selected palette, default F (FALSE); can be set to T (TRUE).
`SingleColour`	a colour hexcode (starting with #), a number between 1-154, or names of colours from `grafify` colour palettes to fill along X-axis aesthetic. Accepts any colour other than "black"; use `grey_lin11`, which is almost black.
`TextXAngle`	orientation of text on X-axis; default 0 degrees. Change to 45 or 90 to remove overlapping text.
`fontsize`	parameter of `base_size` of fonts in `theme_classic`, default set to size 20.

Details

Standard deviation (SD) is plotted through `stat_summary` calculated using `mean_sdl` from the `ggplot2` package (get help with ?mean_sdl), and 1x SD is plotted (`fun.arg = list(mult = 1)`). The X variable is mapped to the `fill` aesthetic in the bar geometry and `colour` aesthetic in `geom_point`.

Colours can be changed using `ColPal`, `ColRev` or `ColSeq` arguments. Colours available can be seen quickly with `plot_grafify_palette`. `ColPal` can be one of the following: "okabe_ito", "dark", "light", "bright", "pale", "vibrant, "muted" or "contrast". `ColRev` (logical TRUE/FALSE) decides whether colours are chosen from first-to-last or last-to-first from within the chosen palette. `ColSeq` (logical TRUE/FALSE) decides whether colours are picked by respecting the order in the palette or the most distant ones using `colorRampPalette`.

If there are many groups along the X axis and you prefer a single colour for the graph,use the `SingleColour` argument.

Three types of plots are available for scatter/jitter symbols and either bars+SD, boxplot or violin plots: `plot_scatterbar_sd`, `plot_scatterbox` and `plot_scatterviolin`.

These are related to the three "dot" versions that use a different geometry for symbols: `plot_dotbox`, `plot_dotbar_sd` and `plot_dotviolin`.

Value

This function returns a `ggplot2` object of class "gg" and "ggplot".

Examples

```
#with jitter
plot_scatterbar_sd(data = data_cholesterol,
xcol = Treatment, ycol = Cholesterol)

#white bars
plot_scatterbar_sd(data = data_cholesterol,
xcol = Treatment, ycol = Cholesterol,
b_alpha = 0)

plot_scatterbar_sd(data = data_doubling_time,
xcol = Student, ycol = Doubling_time,
SingleColour = "ok_grey")
```

`plot_scatterbox` *Plot a scatter plot on a boxplot with two variables.*

Description

This function takes a data table, X and Y variables, and plots a graph with a scatter plot and box and whiskers using `geom_boxplot` and `geom_point` geometries. The boxplot shows IQR and whiskers depict 1.5*IQR. Note that `geom_boxplot` option for outliers is set to `outlier.alpha = 0`. The X variable is mapped to the `fill` aesthetic in both boxplot and symbols, and its colour can be changed using `ColPal` option. Colours can be changed using `ColPal`, `ColRev` or `ColSeq` arguments. Colours available can be seen quickly with `plot_grafify_palette`. `ColPal` can be one of the following: "okabe_ito", "dark", "light", "bright", "pale", "vibrant, "muted" or "contrast". `ColRev` (logical TRUE/FALSE) decides whether colours are chosen from first-to-last or last-to-first from within the chosen palette. `ColSeq` (logical TRUE/FALSE) decides whether colours are picked by respecting the order in the palette or the most distant ones using `colorRampPalette`.

Usage

```
plot_scatterbox(
  data,
  xcol,
  ycol,
  symsize = 2.5,
  symthick = 1,
  jitter = 0.2,
  b_alpha = 1,
  s_alpha = 1,
  ColPal = c("okabe_ito", "all_grafify", "bright", "contrast", "dark", "fishy",
    "kelly", "light", "muted", "pale", "r4", "safe", "vibrant"),
  ColSeq = TRUE,
  ColRev = FALSE,
  SingleColour = "NULL",
  TextXAngle = 0,
  fontsize = 20,
  ...
)
```

Arguments

`data`	a data table object, e.g. data.frame or tibble.

`xcol`	name of the column to plot on X axis. This should be a categorical variable.
`ycol`	name of the column to plot on quantitative Y axis. This should be a quantitative variable.
`symsize`	size of symbols used by `geom_point`. Default set to 2.5, increase/decrease as needed.
`symthick`	thickness of symbol border (`stroke` parameter of `geom_point`), default set to 1.
`jitter`	extent of jitter (scatter) of symbols, default is 0 (i.e. aligned symbols). To reduce symbol overlap, try 0.1-0.3 or higher.
`b_alpha`	fractional opacity of boxplot, default set to 1 (i.e. maximum opacity & zero transparency).
`s_alpha`	fractional opacity of symbols, default set to 1 (i.e. maximum opacity & zero transparency).
`ColPal`	grafify colour palette to apply, default "okabe_ito"; see `graf_palettes` for available palettes.
`ColSeq`	logical TRUE or FALSE. Default TRUE for sequential colours from chosen palette. Set to FALSE for distant colours, which will be applied using `scale_fill_grafify2`.
`ColRev`	whether to reverse order of colour within the selected palette, default F (FALSE); can be set to T (TRUE).
`SingleColour`	a colour hexcode (starting with #), a number between 1-154, or names of colours from `grafify` colour palettes to fill along X-axis aesthetic. Accepts any colour other than "black"; use `grey_lin11`, which is almost black.
`TextXAngle`	orientation of text on X-axis; default 0 degrees. Change to 45 or 90 to remove overlapping text.
`fontsize`	parameter of `base_size` of fonts in `theme_classic`, default set to size 20.
`...`	any additional arguments to pass to `ggplot2`geom_boxplot.

Details

If there are many groups along the X axis and you prefer a single colour for the graph,use the `SingleColour` argument.

The size of symbols can be adjusted using `symsize` set to 1 by default. Transparency of boxplot and symbols can be set independently with `b_alpha` and `s_alpha`, respectively.

Three types of plots are available for scatter/jitter symbols and either bars+SD, boxplot or violin plots: `plot_scatterbar_sd`, `plot_scatterbox` and `plot_scatterviolin`. These are related to the three "dot" versions that use a different geometry for symbols: `plot_scatterbox`, `plot_dotbar_sd` and `plot_dotviolin`.

Value

This function returns a `ggplot2` object of class "gg" and "ggplot".

Examples

```
plot_scatterbox(data = data_cholesterol,
xcol = Treatment, ycol = Cholesterol)

plot_scatterbox(data = data_doubling_time,
xcol = Student, ycol = Doubling_time,
SingleColour = "ok_grey")
```

`plot_scatterviolin` *Plot a scatter plot on a violin plot with two variables.*

Description

This function takes a data table, X and Y variables, and plots a graph with a scatter plot and violinplot using `ggplot`.

Usage

```
plot_scatterviolin(
  data,
  xcol,
  ycol,
  symsize = 2.5,
  symthick = 1,
  bwid = 0.2,
  bvthick = 1,
  b_alpha = 0,
  s_alpha = 1,
  v_alpha = 1,
  ColPal = c("okabe_ito", "all_grafify", "bright", "contrast", "dark", "fishy",
    "kelly", "light", "muted", "pale", "r4", "safe", "vibrant"),
  ColSeq = TRUE,
  ColRev = FALSE,
  SingleColour = "NULL",
  jitter = 0.2,
  trim = TRUE,
  scale = "width",
  TextXAngle = 0,
  fontsize = 20,
  ...
)
```

Arguments

`data`	a data table object, e.g. data.frame or tibble.
`xcol`	name of the column to plot on X axis. This should be a categorical variable.

`ycol`	name of the column to plot on quantitative Y axis. This should be a quantitative variable.
`symsize`	size of dots relative to `binwidth` used by `geom_point`. Default set to 2.5, increase/decrease as needed.
`symthick`	thickness of dot border (`stroke` parameter of `geom_point`), default set to 1.
`bwid`	width of boxplots; default 0.2
`bvthick`	thickness of both violin and boxplot lines; default 1.
`b_alpha`	fractional opacity of boxplots. Default is set to 0, which results in white boxes inside violins. Change to any value >0 up to 1 for different levels of transparency.
`s_alpha`	fractional opacity of symbols, default set to 1 (i.e. maximum opacity & zero transparency). Set `s_alpha = 0` to not show scatter plot.
`v_alpha`	fractional opacity of violins, default set to 1 (i.e. maximum opacity & zero transparency).
`ColPal`	grafify colour palette to apply, default "okabe_ito"; see `graf_palettes` for available palettes.
`ColSeq`	logical TRUE or FALSE. Default TRUE for sequential colours from chosen palette. Set to FALSE for distant colours, which will be applied using `scale_fill_grafify2`.
`ColRev`	whether to reverse order of colour within the selected palette, default F (FALSE); can be set to T (TRUE).
`SingleColour`	a colour hexcode (starting with #), a number between 1-154, or names of colours from `grafify` colour palettes to fill along X-axis aesthetic. Accepts any colour other than "black"; use `grey_lin11`, which is almost black.
`jitter`	extent of jitter (scatter) of symbols, default is 0 (i.e. aligned symbols). To reduce symbol overlap, try 0.1-0.3 or higher.
`trim`	set whether tips of violin plot should be trimmed at high/low data. Default `trim = T`, can be changed to F.
`scale`	set to "area" by default, can be changed to "count" or "width".
`TextXAngle`	orientation of text on X-axis; default 0 degrees. Change to 45 or 90 to remove overlapping text.
`fontsize`	parameter of `base_size` of fonts in `theme_classic`, default set to size 20.
`...`	any additional arguments to pass to `ggplot2`geom_boxplot, `ggplot2`geom_point or `ggplot2`geom_violin.

Details

The function uses `geom_violin`, `geom_boxplot` and `geom_point` geometries. Note that the `geom_violin` options are set as follows: `scale = "width"`. The `trim = T` set by default can be changed when calling the function. The boxplot shows IQR and the median is marked with a thicker horizontal line, and whisker depicts 1.5*IQR. The X variable is mapped to the `fill` aesthetic in both violin and symbols, and its colour can be changed using `ColPal` option. Colours can be changed using `ColPal`, `ColRev` or `ColSeq` arguments. Colours available can be seen quickly with `plot_grafify_palette`. `ColPal` can be one of the following: "okabe_ito", "dark", "light", "bright", "pale", "vibrant, "muted" or "contrast". `ColRev` (logical TRUE/FALSE) decides whether colours are chosen from first-to-last or last-to-first from within the chosen palette. `ColSeq` (logical TRUE/FALSE) decides whether colours are picked by respecting the order in the palette or the most distant ones using `colorRampPalette`.

If there are many groups along the X axis and you prefer a single colour for the graph,use the `SingleColour` argument.

The size of symbols can be adjusted using `symsize` set to 1 by default. Transparency of violins and symbols can be set independently with `v_alpha` and `s_alpha`, respectively.

Three types of plots are available for scatter/jitter symbols and either bars+SD, boxplot or violin plots: `plot_scatterbar_sd`, `plot_scatterbox` and `plot_scatterviolin`. These are related to the three "dot" versions that use a different geometry for symbols: `plot_dotbox`, `plot_dotbar_sd` and `plot_dotviolin`.

Value

This function returns a `ggplot2` object of class "gg" and "ggplot".

Examples

```
#plot without jitter
plot_scatterviolin(data = data_t_pdiff,
xcol = Condition, ycol = Mass,
symsize = 2, trim = FALSE)

#no symbols
plot_scatterviolin(data = data_t_pdiff,
xcol = Condition, ycol = Mass,
s_alpha = 0,
symsize = 2, trim = FALSE)

#single colour along X
plot_scatterviolin(data = data_t_pdiff,
xcol = Condition, ycol = Mass,
SingleColour = "pale_blue",
s_alpha = 0,
symsize = 2, trim = FALSE)
```

`plot_xy_CatGroup` *Plot points on a quantitative X - Y plot & a categorical grouping variable.*

Description

This function takes a data table, quantitative X and Y variables along with a categorical grouping variable, and a and plots a graph with using `geom_point`. The categorical `CatGroup` variable is mapped to the `fill` aesthetic of symbols.

Usage

```
plot_xy_CatGroup(
  data,
  xcol,
  ycol,
  CatGroup,
  symsize = 2.5,
  symthick = 1,
  s_alpha = 1,
  ColPal = c("okabe_ito", "all_grafify", "bright", "contrast", "dark", "fishy",
    "kelly", "light", "muted", "pale", "r4", "safe", "vibrant"),
  ColSeq = TRUE,
  ColRev = FALSE,
  TextXAngle = 0,
  fontsize = 20,
  ...
)
```

Arguments

`data`	a data table object, e.g. data.frame or tibble.
`xcol`	name of the column with quantitative X variable.
`ycol`	name of the column with quantitative Y variable.
`CatGroup`	a categorical variable as grouping factor for colour of data points, should be a categorical variable for default colours to work. Will be converted to `factor` if your column is numeric
`symsize`	size of symbols used by `geom_point`. Default set to 2.5, increase/decrease as needed.
`symthick`	thickness of symbol border, default set to 1.
`s_alpha`	fractional opacity of symbols, default set to 1 (i.e. maximum opacity & zero transparency).
`ColPal`	grafify colour palette to apply, default "okabe_ito"; see `graf_palettes` for available palettes.
`ColSeq`	logical TRUE or FALSE. Default TRUE for sequential colours from chosen palette. Set to FALSE for distant colours, which will be applied using `scale_fill_grafify2`.
`ColRev`	whether to reverse order of colour within the selected palette, default F (FALSE); can be set to T (TRUE).
`TextXAngle`	orientation of text on X-axis; default 0 degrees. Change to 45 or 90 to remove overlapping text.
`fontsize`	parameter of `base_size` of fonts in `theme_classic`, default set to size 20.
`...`	any additional arguments to pass to `ggplot2`geom_point.

Details

Colours can be changed using `ColPal`, `ColRev` or `ColSeq` arguments. Colours available can be seen quickly with `plot_grafify_palette`. `ColPal` can be one of the following: "okabe_ito", "dark", "light", "bright", "pale", "vibrant, "muted" or "contrast". `ColRev` (logical TRUE/FALSE) decides whether colours are chosen from first-to-last or last-to-first from within the chosen palette. `ColSeq` (logical TRUE/FALSE) decides whether colours are picked by respecting the order in the palette or the most distant ones using `colorRampPalette`.

This plot is related to `plot_xy_NumGroup` which requires a numeric grouping factor. When summary statistics (mean/median) are required, use `plot_3d_scatterbar`, `plot_3d_scatterbox` or `plot_4d_scatterbox`.

Value

This function returns a `ggplot2` object of class "gg" and "ggplot".

Examples

```
#The grouping factor cyl  is automatically converted to categorical variable
plot_xy_CatGroup(data = mtcars,
xcol = mpg, ycol = disp, CatGroup = cyl,
ColPal = "vibrant", ColSeq = FALSE)
```

`plot_xy_NumGroup` *Plot points on a quantitative X - Y plot & a numeric grouping variable.*

Description

This function takes a data table, quantitative X and Y variables, and a numeric grouping variable, and a and plots a graph with using `geom_point`. The numerical `NumGroup` variable is mapped to the `fill` aesthetic of symbols, which receives the `scale_fill_grafify` default quantitative palette (`blue_conti`).

Usage

```
plot_xy_NumGroup(
  data,
  xcol,
  ycol,
  NumGroup,
  symsize = 2.5,
  symthick = 1,
  s_alpha = 1,
 ColPal = c("blue_conti", "yellow_conti", "grey_conti", "PrGn_div", "OrBl_div"),
  TextXAngle = 0,
  fontsize = 20,
  ...
)
```

Arguments

`data`	a data table object, e.g. data.frame or tibble.
`xcol`	name of the column with quantitative X variable.
`ycol`	name of the column with quantitative Y variable.
`NumGroup`	a numeric factor for `fill` aesthetic of data points.
`symsize`	size of symbols used by `geom_point`. Default set to 2.5, increase/decrease as needed.
`symthick`	thickness of symbol border, default set to 1.
`s_alpha`	fractional opacity of symbols, default set to 1 (i.e. maximum opacity & zero transparency).
`ColPal`	One of five quantitative colour schemes in `grafify`: `blue_conti`, `yellow_conti`, `grey_conti`, `PrGn_div`, `OrBl_div`.
`TextXAngle`	orientation of text on X-axis; default 0 degrees. Change to 45 or 90 to remove overlapping text.
`fontsize`	parameter of `base_size` of fonts in `theme_classic`, default set to size 20.
`...`	any additional arguments to pass to `ggplot2`geom_violin.

Details

This plot is related to `plot_xy_CatGroup` which requires a categorical grouping factor. When summary statistics (mean/median) are required, use `plot_3d_scatterbar`, `plot_3d_scatterbox` or `plot_4d_scatterbox`.

Value

This function returns a `ggplot2` object of class "gg" and "ggplot".

Examples

```
#The grouping factor gear is numeric
plot_xy_NumGroup(data = mtcars,
xcol = mpg, ycol = disp, NumGroup = cyl,
s_alpha = 0.8)
#change colour palette
plot_xy_NumGroup(data = mtcars,
xcol = mpg, ycol = disp, NumGroup = cyl,
s_alpha = 0.8,
ColPal = "grey_conti")
```

`posthoc_Levelwise` *Level-wise post-hoc comparisons from a linear or linear mixed effects model.*

Description

This function is a wrapper based on `emmeans`, and needs a ordinary linear model produced by `simple_model` or a mixed effects model produced by `mixed_model` or `mixed_model_slopes` (or generated directly with `lm`, `lme4` or `lmerTest` calls). It also needs to know the fixed factor(s), which should match those in the model and data table.

Usage

```
posthoc_Levelwise(Model, Fixed_Factor, P_Adj = "fdr", Factor, ...)
```

Arguments

`Model`	a model object fit using `simple_model` or `mixed_model` or related.
`Fixed_Factor`	one or more categorical variables, provided as a vector (see Examples), whose levels you wish to compare pairwise. Names of Fixed_Factor should match Fixed_Factor used to fit the model. When more than one factor is provided e.g. `Fixed_factor = c("A", "B")`, this function passes this on as `specs = A|B` (note the vertical | between the two Fixed_Factor) to `emmeans` to produce comparisons between each level A with each other listed separately at each level of B.
`P_Adj`	method for correcting P values for multiple comparisons. Default is set to false discovery rate ("fdr"), can be changed to "none", "tukey", "bonferroni", "sidak". See Interaction analysis in emmeans in the manual for `emmeans`.
`Factor`	old argument name for `Fixed_Factor`; retained for backward compatibility.
`...`	additional arguments for `emmeans` such as `lmer.df` or others. See help for sophisticated models in emmeans.

Details

The function will generate level-wise comparisons (as described in Comparisons and contrasts in emmeans), i.e. comparison between of every level of one factor separately at each level of the other factor. By default, P values are corrected by the FDR method (which can be changed). If the model was fit by transforming the quantitative response variable using "log", "logit", "sqrt" etc., results will still be on the original scale, i.e. `type = "response"` is the default; data will be back-transformed (check results to confirm this), and for log or logit see Transformations and link functions in emmeans, ratios will be compared. The first part of the `emmeans` results has the estimated marginal means, SE and CI (`$emmeans`), which are generated from the fitted model, and **not** the original data table. The second part has the results of the comparisons (`$contrasts`).

Value

returns an "emm_list" object containing contrasts and emmeans through `emmeans`.

Examples

```
#make a linear model first
CholMod <- mixed_model(data = data_cholesterol,
Y_value =  "Cholesterol",
Fixed_Factor = c("Hospital", "Treatment"),
Random_Factor =  "Subject")

#note quotes used only for fixed Fixed_Factor
#to get comparisons between different hospitals separately for each level of Treatment
posthoc_Levelwise(Model = CholMod,
Fixed_Factor = c("Hospital", "Treatment"))

#get comparisons between treatments separately at each hospital
posthoc_Levelwise(Model = CholMod,
Fixed_Factor = c("Treatment", "Hospital"))
```

posthoc_Pairwise *Pairwise post-hoc comparisons from a linear or linear mixed effects model.*

Description

This function is a wrapper based on `emmeans`, and needs a ordinary linear model produced by `simple_model` or a mixed effects model produced by `mixed_model` or `mixed_model_slopes` (or generated directly with `lm`, `lme4` or `lmerTest` calls). It also needs to know the fixed factor(s), which should match those in the model and data table.

Usage

```
posthoc_Pairwise(Model, Fixed_Factor, P_Adj = "fdr", Factor, ...)
```

Arguments

`Model`	a model object fit using `simple_model` or `mixed_model` or related.
`Fixed_Factor`	one or more categorical variables, provided as a vector (see Examples), whose levels you wish to compare pairwise. Names of Fixed_Factor should match Fixed_Factor used to fit the model. When more than one factor is provided e.g. `Fixed_factor = c("A", "B")`, this function passes this on as `specs = A:B` (note the colon between the two Fixed_Factor) to `emmeans` to produce pairwise comparisons.
`P_Adj`	method for correcting P values for multiple comparisons. Default is set to false discovery rate ("fdr"), can be changed to "none", "tukey", "bonferroni", "sidak". See Interaction analysis in emmeans in the manual for `emmeans`.
`Factor`	old argument name for `Fixed_Factor`; retained for backward compatibility.
`...`	additional arguments for `emmeans` such as `lmer.df` or others. See help for sophisticated models in emmeans.

Details

The function will generate pairwise comparisons of every level of every factor (as described in Comparisons and contrasts in emmeans). Too many comparisons will be generated and only use this when necessary. By default, P values are corrected by the FDR method (which can be changed). If the model was fit by transforming the quantitative response variable using "log", "logit", "sqrt" etc., results will still be on the original scale, i.e. `type = "response"` is the default; data will be back-transformed (check results to confirm this), and for log or logit see Transformations and link functions in emmeans, ratios will be compared. The first part of the `emmeans` results has the estimated marginal means, SE and CI (`$emmeans`), which are generated from the fitted model, and **not** the original data table. The second part has the results of the comparisons (`$contrasts`).

Value

returns an "emm_list" object containing contrasts and emmeans through `emmeans`.

Examples

```
#make linear models first
DoublMod <- simple_model(data = data_doubling_time,
Y_value =  "Doubling_time", Fixed_Factor = "Student")
CholMod <- mixed_model(data = data_cholesterol,
Y_value = "Cholesterol",
Fixed_Factor = c("Hospital", "Treatment"),
Random_Factor = "Subject")

posthoc_Pairwise(Model = DoublMod,
Fixed_Factor = "Student")

#basic use with two Fixed_Factor provided as a vector
posthoc_Pairwise(Model = CholMod,
Fixed_Factor = c("Treatment", "Hospital"))

#same call with "tukey" adjustment
posthoc_Pairwise(Model = CholMod,
Fixed_Factor = c("Treatment", "Hospital"),
P_adj = "tukey")
```

`posthoc_Trends_Levelwise` *Use emtrends to get level-wise comparison of slopes from a linear model.*

Description

This function is a wrapper based on `emmeans`, and needs a ordinary linear model produced by `simple_model` or a mixed effects model produced by `mixed_model` or `mixed_model_slopes` (or generated directly with `lm`, `lme4` or `lmerTest` calls). At least one of the factors should be a numeric covariate whose slopes you wish to find. It also needs to know the fixed factor(s), which should match those in the model and data table.

Usage

```
posthoc_Trends_Levelwise(
  Model,
  Fixed_Factor,
  Trend_Factor,
  P_Adj = "sidak",
  ...
)
```

Arguments

`Model`	a model object fit using `simple_model` or `mixed_model` (or `lm` or `lmer`).
`Fixed_Factor`	one or more categorical variables, provided as a vector (see Examples), whose levels you wish to compare pairwise. Names of Fixed_Factor should match Fixed_Factor used to fit the model. When more than one factor is provided e.g. `Fixed_factor = c("A", "B")`, this function passes this on as `specs = A:B` (note the colon between the two Fixed_Factor) to `emmeans` to produce pairwise comparisons.
`Trend_Factor`	a quantitative variable that interacts with a factor and whose slope (trend) is to be compared
`P_Adj`	method for correcting P values for multiple comparisons. Default is "sidak", can be changed to "bonferroni". See Interaction analysis in emmeans in the manual for `emmeans`.
`...`	additional arguments for `emmeans` such as `lmer.df` or others. See help for sophisticated models in emmeans.

Details

Checkout the Interactions with covariates section in the emmeans vignette for more details. One of the independent variables should be a quantitative (e.g. time points) variable whose slope (trend) you want to find at levels of the other factor.

Value

returns an "emm_list" object containing slopes and their contrasts calculated through `emtrends`.

Examples

```
#create an lm model
#Time2 is numeric (time points)
m1 <- simple_model(data = data_2w_Tdeath,
Y_value = "PI", Fixed_Factor = c("Genotype", "Time2"))
posthoc_Trends_Levelwise(Model = m1,
Fixed_Factor = "Genotype",
Trend_Factor = "Time2")
```

posthoc_Trends_Pairwise

Use emtrends to get pairwise comparison of slopes from a linear model.

Description

This function is a wrapper based on `emmeans`, and needs a ordinary linear model produced by `simple_model` or a mixed effects model produced by `mixed_model` or `mixed_model_slopes` (or generated directly with `lm`, `lme4` or `lmerTest` calls). At least one of the factors should be a numeric covariate whose slopes you wish to find. It also needs to know the fixed factor(s), which should match those in the model and data table.

Usage

```
posthoc_Trends_Pairwise(
  Model,
  Fixed_Factor,
  Trend_Factor,
  P_Adj = "sidak",
  ...
)
```

Arguments

`Model`	a model object fit using `simple_model` or `mixed_model` (or `lm` or `lmer`).
`Fixed_Factor`	one or more categorical variables, provided as a vector (see Examples), whose levels you wish to compare pairwise. Names of Fixed_Factor should match Fixed_Factor used to fit the model. When more than one factor is provided e.g. `Fixed_factor = c("A", "B")`, this function passes this on as `specs = A:B` (note the colon between the two Fixed_Factor) to `emmeans` to produce pairwise comparisons.
`Trend_Factor`	a quantitative variable that interacts with a factor and whose slope (trend) is to be compared
`P_Adj`	method for correcting P values for multiple comparisons. Default is "sidak", can be changed to "bonferroni". See Interaction analysis in emmeans in the manual for `emmeans`.
`...`	additional arguments for `emmeans` such as `lmer.df` or others. See help for sophisticated models in emmeans.

Details

Checkout the Interactions with covariates section in the emmeans vignette for more details. One of the independent variables should be a quantitative (e.g. time points) variable whose slope (trend) you want to find at levels of the other factor.

Value

returns an "emm_list" object containing slopes and their contrasts calculated through `emtrends`.

Examples

```
#create an lm model
#Time2 is numeric (time points)
m1 <- simple_model(data = data_2w_Tdeath,
Y_value = "PI", Fixed_Factor = c("Genotype", "Time2"))
posthoc_Trends_Pairwise(Model = m1,
Fixed_Factor = "Genotype",
Trend_Factor = "Time2")
```

`posthoc_Trends_vsRef` *Use emtrends to get level-wise comparison of slopes from a linear model.*

Description

This function is a wrapper based on `emmeans`, and needs a ordinary linear model produced by `simple_model` or a mixed effects model produced by `mixed_model` or `mixed_model_slopes` (or generated directly with `lm`, `lme4` or `lmerTest` calls). At least one of the factors should be a numeric covariate whose slopes you wish to find. It also needs to know the fixed factor(s), which should match those in the model and data table.

Usage

```
posthoc_Trends_vsRef(
  Model,
  Fixed_Factor,
  Trend_Factor,
  Ref_Level = 1,
  P_Adj = "sidak",
  ...
)
```

Arguments

`Model`	a model object fit using `simple_model` or `mixed_model` (or `lm` or `lmer`).
`Fixed_Factor`	one or more categorical variables, provided as a vector (see Examples), whose levels you wish to compare pairwise. Names of Fixed_Factor should match Fixed_Factor used to fit the model. When more than one factor is provided e.g. `Fixed_factor = c("A", "B")`, this function passes this on as `specs = A:B` (note the colon between the two Fixed_Factor) to `emmeans` to produce pairwise comparisons.

`Trend_Factor`	a quantitative variable that interacts with a factor and whose slope (trend) is to be compared
`Ref_Level`	the level within that factor to be considered the reference or control to compare other levels to (to be provided as a number - by default R orders levels alphabetically); default `Ref_Level = 1`.
`P_Adj`	method for correcting P values for multiple comparisons. Default is "sidak", can be changed to "bonferroni". See Interaction analysis in emmeans in the manual for `emmeans`.
`...`	additional arguments for `emmeans` such as `lmer.df` or others. See help for sophisticated models in emmeans.

Details

Checkout the Interactions with covariates section in the emmeans vignette for more details. One of the independent variables should be a quantitative (e.g. time points) variable whose slope (trend) you want to find at levels of the other factor.

Value

returns an "emm_list" object containing slopes and their contrasts calculated through `emtrends`.

Examples

```
#create an lm model
#Time2 is numeric (time points)
m1 <- simple_model(data = data_2w_Tdeath,
Y_value = "PI", Fixed_Factor = c("Genotype", "Time2"))
posthoc_Trends_vsRef(Model = m1,
Fixed_Factor = "Genotype",
Trend_Factor = "Time2",
Ref_Level = 2)
```

`posthoc_vsRef` *Post-hoc comparisons to a control or reference group.*

Description

This function is a wrapper based on `emmeans`, and needs a ordinary linear model produced by `simple_model` or a mixed effects model produced by `mixed_model` or `mixed_model_slopes` (or generated directly with `lm`, `lme4` or `lmerTest` calls). It also needs to know the fixed factor(s), which should match those in the model and data table.

Usage

```
posthoc_vsRef(Model, Fixed_Factor, Ref_Level = 1, P_Adj = "fdr", Factor, ...)
```

Arguments

`Model`	a model object fit using `simple_model` or `mixed_model` or related.
`Fixed_Factor`	Fixed_Factor one or more categorical variables, provided as a vector (see Examples), whose levels you wish to compare pairwise. Names of Fixed_Factor should match Fixed_Factor used to fit the model. When more than one factor is provided e.g. `Fixed_factor = c("A", "B")`, this function passes this on as `specs = A|B` (note the vertical | between the two Fixed_Factor) to `emmeans`. The specification internally is set to `specs = trt.vs.ctrl, Ref_Level = 1` to compare each group in A to the reference first group in A, separately at each level of B.
`Ref_Level`	the level within that factor to be considered the reference or control to compare other levels to (to be provided as a number - by default R orders levels alphabetically); default `Ref_Level = 1`.
`P_Adj`	method for correcting P values for multiple comparisons. Default is set to false discovery rate ("fdr"), can be changed to "none", "tukey", "bonferroni", "sidak". See Interaction analysis in emmeans in the manual for `emmeans`.
`Factor`	old argument name for `Fixed_Factor`; retained for backward compatibility.
`...`	additional arguments for `emmeans` such as `lmer.df` or others. See help for sophisticated models in emmeans.

Details

The function will generate treatment vs control type of comparisons (as described in Comparisons and contrasts in emmeans), i.e. comparison of each level of a factor to a reference level, which is set by default to the first level in the factor (`Ref_Level = 1`). By default, P values are corrected by the FDR method (which can be changed). If the model was fit by transforming the quantitative response variable using "log", "logit", "sqrt" etc., results will still be on the original scale, i.e. `type = "response"` is the default; data will be back-transformed (check results to confirm this), and for log or logit see Transformations and link functions in emmeans, ratios will be compared. The first part of the `emmeans` results has the estimated marginal means, SE and CI (`$emmeans`), which are generated from the fitted model, and **not** the original data table. The second part has the results of the comparisons (`$contrasts`).

Value

returns an "emm_list" object containing contrasts and emmeans through `emmeans`.

Examples

```
#make linear models first
DoublMod <- simple_model(data = data_doubling_time,
Y_value =  "Doubling_time",
Fixed_Factor =  "Student")

CholMod <- mixed_model(data = data_cholesterol,
Y_value =  "Cholesterol",
Fixed_Factor = c("Hospital", "Treatment"),
Random_Factor = "Subject")
```

```
#to compare all students with student #9
posthoc_vsRef(Model = DoublMod,
Fixed_Factor = "Student", Ref_Level = 9)

#for comparison between hospital_a to every other hospital, separately at levels of Treatment
posthoc_vsRef(Model = CholMod,
Fixed_Factor = c("Hospital", "Treatment"), Ref_Level = 1)
```

`scale_color_grafify` `scale_colour_` *and* `scale_fill_` *functions*

Description

These let you apply `grafify` discrete or continuous palettes as fill or colour/color aesthetics to any `ggplot2` (`scale_color_` spelling is also accepted).

Usage

```
scale_color_grafify(
  palette = "okabe_ito",
  discrete = TRUE,
  ColSeq = TRUE,
  reverse = FALSE,
  ...
)
```

Arguments

`palette`	Name of the `grafify` palettes from above, provide within quotes. Default discrete palette is `okabe_ito`. For quantitative palette, set `discrete = FALSE` (which will apply `blue_conti` unless another palette is chosen).
`discrete`	logical TRUE or FALSE. Default is TRUE for discrete colour palettes. Set to FALSE when plotting quantitative data to use quantitative palettes from above.
`ColSeq`	logical TRUE or FALSE. Default TRUE for sequential colours from chosen palette. Set to FALSE for distant colours.
`reverse`	Whether the colour order should be reversed.
`...`	Additional parameters for `scale_fill` or `scale_colour`.

Details

The default is `palette = "okabe_ito"` and `discrete = TRUE`. To apply quantitative colour/fill schemes set `discrete = FALSE` (if a palette is not chosen, `blue_conti` palette will be applied by default).

Categorical/discreet palettes (only allowed with `discrete = TRUE`)

- `okabe_ito`

- `bright`
- `contrast`
- `dark`
- `kelly`
- `light`
- `muted`
- `pale`
- `r4`
- `safe`
- `vibrant`

By default, sequential colours from above palettes will be chosen. To choose the most distant colours set `ColSeq = TRUE`.

Sequential quantitative palettes (only allowed with `discrete = FALSE`):

- `grey_conti`
- `blue_conti`
- `yellow_conti`

Divergent quantitative palettes (only allowed with `discrete = FALSE`):

- `OrBl_div`
- `PrGn_div`

Value

ggplot scale_fill function for discrete colours.

Examples

```
#add a grafify fill scheme to ggplot
ggplot(emmeans::neuralgia, aes(x = Treatment,
                               y = Duration))+
  geom_boxplot(aes(fill = Treatment),
               alpha = .6)+
  geom_point(aes(colour = Treatment,
                 shape = Treatment),
             size = 3)+
  scale_fill_grafify(palette = "bright")+
  scale_colour_grafify(palette = "bright")+
  facet_wrap("Sex")+
  theme_classic()
#distant colours `ColSeq = FALSE`
ggplot(emmeans::neuralgia, aes(x = Treatment,
                               y = Duration))+
  geom_boxplot(aes(fill = Treatment),
               alpha = .6)+
  geom_point(aes(colour = Treatment,
                 shape = Treatment),
             size = 3)+
  scale_fill_grafify(palette = "bright",
                     ColSeq = FALSE)+
  scale_colour_grafify(palette = "bright",
                       ColSeq = FALSE)+
  facet_wrap("Sex")+
  theme_classic()
#quantiative colour schemes with `discrete = FALSE`
ggplot(mtcars, aes(x = disp,
                   y = mpg))+
  geom_point(aes(colour = cyl),
             size = 3)+
  scale_colour_grafify(discrete = FALSE)
#quantiative colour scheme
ggplot(mtcars, aes(x = disp,
                   y = mpg))+
  geom_point(aes(colour = cyl),
             size = 3)+
  scale_colour_grafify(palette = "yellow_conti")
```

---

scale_colour_grafify   scale_colour_ *and* scale_fill_ *functions*

Description

These let you apply grafify discrete or continuous palettes as fill or colour/color aesthetics to any ggplot2 (scale_color_ spelling is also accepted).

Usage

```
scale_colour_grafify(
  palette = "okabe_ito",
  discrete = TRUE,
  ColSeq = TRUE,
  reverse = FALSE,
  ...
)
```

Arguments

palette	Name of the grafify palettes from above, provide within quotes. Default discrete palette is okabe_ito. For quantitative palette, set discrete = FALSE (which will apply blue_conti unless another palette is chosen).
discrete	logical TRUE or FALSE. Default is TRUE for discrete colour palettes. Set to FALSE when plotting quantitative data to use quantitative palettes from above.
ColSeq	logical TRUE or FALSE. Default TRUE for sequential colours from chosen palette. Set to FALSE for distant colours.

`reverse`	Whether the colour order should be reversed.
`...`	Additional parameters for `scale_fill` or `scale_colour`.

Details

The default is `palette = "okabe_ito"` and `discrete = TRUE`. To apply quantitative colour/fill schemes set `discrete = FALSE` (if a palette is not chosen, `blue_conti` palette will be applied by default).

Categorical/discreet palettes (only allowed with `discrete = TRUE`)

- `okabe_ito`
- `bright`
- `contrast`
- `dark`
- `kelly`
- `light`
- `muted`
- `pale`
- `r4`
- `safe`
- `vibrant`

By default, sequential colours from above palettes will be chosen. To choose the most distant colours set `ColSeq = TRUE`.

Sequential quantitative palettes (only allowed with `discrete = FALSE`):

- `grey_conti`
- `blue_conti`
- `yellow_conti`

Divergent quantitative palettes (only allowed with `discrete = FALSE`):

- `OrBl_div`
- `PrGn_div`

Value

ggplot scale_fill function for discrete colours.

Examples

```
#add a grafify fill scheme to ggplot
ggplot(emmeans::neuralgia, aes(x = Treatment,
                               y = Duration))+
  geom_boxplot(aes(fill = Treatment),
               alpha = .6)+
  geom_point(aes(colour = Treatment,
                 shape = Treatment),
             size = 3)+
  scale_fill_grafify(palette = "bright")+
  scale_colour_grafify(palette = "bright")+
  facet_wrap("Sex")+
  theme_classic()
#distant colours `ColSeq = FALSE`
ggplot(emmeans::neuralgia, aes(x = Treatment,
                               y = Duration))+
  geom_boxplot(aes(fill = Treatment),
               alpha = .6)+
  geom_point(aes(colour = Treatment,
                 shape = Treatment),
             size = 3)+
  scale_fill_grafify(palette = "bright",
                     ColSeq = FALSE)+
  scale_colour_grafify(palette = "bright",
                       ColSeq = FALSE)+
  facet_wrap("Sex")+
  theme_classic()
#quantiative colour schemes with `discrete = FALSE`
ggplot(mtcars, aes(x = disp,
                   y = mpg))+
  geom_point(aes(colour = cyl),
             size = 3)+
  scale_colour_grafify(discrete = FALSE)
#quantiative colour scheme
ggplot(mtcars, aes(x = disp,
                   y = mpg))+
  geom_point(aes(colour = cyl),
             size = 3)+
  scale_colour_grafify(palette = "yellow_conti")
```

`scale_fill_grafify` `scale_colour_` *and* `scale_fill_` *functions*

Description

These let you apply `grafify` discrete or continuous palettes as fill or colour/color aesthetics to any `ggplot2` (`scale_color_` spelling is also accepted).

Usage

```
scale_fill_grafify(
  palette = "okabe_ito",
  discrete = TRUE,
  ColSeq = TRUE,
  reverse = FALSE,
  ...
)
```

Arguments

`palette`	Name of the `grafify` palettes from above, provide within quotes. Default discrete palette is `okabe_ito`. For quantitative palette, set `discrete = FALSE` (which will apply `blue_conti` unless another palette is chosen).
`discrete`	logical TRUE or FALSE. Default is TRUE for discrete colour palettes. Set to FALSE when plotting quantitative data to use quantitative palettes from above.
`ColSeq`	logical TRUE or FALSE. Default TRUE for sequential colours from chosen palette. Set to FALSE for distant colours.
`reverse`	Whether the colour order should be reversed.
`...`	Additional parameters for `scale_fill` or `scale_colour`.

Details

The default is `palette = "okabe_ito"` and `discrete = TRUE`. To apply quantitative colour/fill schemes set `discrete = FALSE` (if a palette is not chosen, `blue_conti` palette will be applied by default).

Categorical/discreet palettes (only allowed with `discrete = TRUE`)

- `okabe_ito`
- `bright`
- `contrast`
- `dark`
- `kelly`
- `light`
- `muted`
- `pale`
- `r4`
- `safe`
- `vibrant`

By default, sequential colours from above palettes will be chosen. To choose the most distant colours set `ColSeq = TRUE`.

Sequential quantitative palettes (only allowed with `discrete = FALSE`):

- `grey_conti`
- `blue_conti`
- `yellow_conti`

Divergent quantitative palettes (only allowed with `discrete = FALSE`):

- `OrBl_div`
- `PrGn_div`

Value

ggplot scale_fill function for discrete colours.

Examples

```
#add a grafify fill scheme to ggplot
ggplot(emmeans::neuralgia, aes(x = Treatment,
                               y = Duration))+
  geom_boxplot(aes(fill = Treatment),
               alpha = .6)+
  geom_point(aes(colour = Treatment,
                 shape = Treatment),
             size = 3)+
  scale_fill_grafify(palette = "bright")+
  scale_colour_grafify(palette = "bright")+
  facet_wrap("Sex")+
  theme_classic()
#distant colours `ColSeq = FALSE`
ggplot(emmeans::neuralgia, aes(x = Treatment,
                               y = Duration))+
  geom_boxplot(aes(fill = Treatment),
               alpha = .6)+
  geom_point(aes(colour = Treatment,
                 shape = Treatment),
             size = 3)+
  scale_fill_grafify(palette = "bright",
                     ColSeq = FALSE)+
  scale_colour_grafify(palette = "bright",
                       ColSeq = FALSE)+
  facet_wrap("Sex")+
  theme_classic()
#quantiative colour schemes with `discrete = FALSE`
ggplot(mtcars, aes(x = disp,
                   y = mpg))+
  geom_point(aes(colour = cyl),
             size = 3)+
  scale_colour_grafify(discrete = FALSE)
#quantiative colour scheme
ggplot(mtcars, aes(x = disp,
                   y = mpg))+
  geom_point(aes(colour = cyl),
             size = 3)+
  scale_colour_grafify(palette = "yellow_conti")
```

`simple_anova` *ANOVA table from a linear model fit to data.*

Description

Update in v0.2.1: This function uses `lm` to fit a linear model to data, passes it on to `Anova`, and outputs the ANOVA table with type II sum of squares with F statistics and *P* values. (Previous versions produced type I sum of squares using `anova` call.)

Usage

```
simple_anova(data, Y_value, Fixed_Factor, ...)
```

Arguments

`data`	a data table object, e.g. data.frame or tibble.
`Y_value`	name of column containing quantitative (dependent) variable, provided within "quotes".
`Fixed_Factor`	name(s) of categorical fixed factors (independent variables) provided as a vector if more than one or within "quotes".
`...`	any additional argument to pass on to `lm` if required.

Details

It requires a data table, one quantitative dependent variable and one or more independent variables. If your experiment design has random factors, use the related function `mixed_anova`.

This function is related to `link{simple_model}`.

Value

ANOVA table of class "anova" and "data.frame".

Examples

```
#Basic usage
simple_anova(data = data_doubling_time,
Y_value = "Doubling_time",
Fixed_Factor = "Student")
```

`simple_model` *Model from a linear model fit to data.*

Description

This function uses `lm` to fit a linear model to data and outputs the model object. It requires a data table, one quantitative dependent variable and one or more independent variables. The model output can be used to extract coefficients and other information, including post-hoc comparisons. If your experiment design has random factors, use the related function `mixed_model`.

Usage

```
simple_model(data, Y_value, Fixed_Factor, ...)
```

Arguments

`data`	a data table object, e.g. data.frame or tibble.
`Y_value`	name of column containing quantitative (dependent) variable, provided within "quotes".
`Fixed_Factor`	name(s) of categorical fixed factors (independent variables) provided as a vector if more than one or within "quotes".
`...`	any additional arguments to pass on to `lm` if required.

Details

This function is related to `link{simple_anova}`. Output of this function can be used with `posthoc_Pairwise`, `posthoc_Levelwise` and `posthoc_vsRef`, or with `emmeans`.

Value

This function returns an object of class "lm".

Examples

```
#fixed factors provided as a vector
Doubmodel <- simple_model(data = data_doubling_time,
Y_value =  "Doubling_time",
Fixed_Factor = "Student")
#get summary
summary(Doubmodel)
```

Index