Package ‘biometryassist’

June 5, 2024

Type Package

Title Functions to Assist Design and Analysis of Agronomic Experiments

Version 1.2.1

Description Provides functions to aid in the design and analysis of agronomic and agricultural experiments through easy access to documentation and helper functions, especially for users who are learning these concepts. While not required for most functionality, this package enhances the `asreml` package which provides a computationally efficient algorithm for fitting mixed models using Residual Maximum Likelihood. It is a commercial package that can be purchased as 'asreml-R' from 'VSNi' <https://vsni.co.uk/>, who will supply a zip file for local installation/updating (see <https://asreml.kb.vsni.co.uk/>).

License MIT + file LICENSE

URL https://biometryhub.github.io/biometryassist/

BugReports https://github.com/biometryhub/biometryassist/issues

Depends R (>= 4.0.0)

Imports agricolae, askpass, cowplot, curl, emmeans, farver, ggplot2, lattice, multcompView, pracma, rlang (>= 1.0.0), scales, stringi, xml2

Suggests covr, crayon, knitr, rmarkdown, testthat, vdiffr, withr

Enhances asreml, ARTool, lme4, nlme, sommer

Config/testthat/edition 3

Config/testthat/parallel true

Encoding UTF-8

RoxygenNote 7.3.1

VignetteBuilder knitr

NeedsCompilation no

Author Sharon Nielsen [aut],
Sam Rogers [aut, cre],
Annie Conway [aut],
University of Adelaide [cph, fnd] (https://adelaide.edu.au/),
Grains Research and Development Corporation [cph, fnd]
(https://grdc.com.au/)

Maintainer Sam Rogers <biometrytraining@adelaide.edu.au>

Repository CRAN

Date/Publication 2024-06-05 15:50:02 UTC

Contents

`autoplot` *Generate automatic plots for objects generated in biometryassist*

Description

Generate automatic plots for objects generated in biometryassist

Usage

```
autoplot(object, ...)

## S3 method for class 'mct'
autoplot(
  object,
  size = 4,
  label_height = 0.1,
  rotation = 0,
  axis_rotation = rotation,
  label_rotation = rotation,
  ...
)

## S3 method for class 'design'
autoplot(
  object,
  rotation = 0,
  size = 4,
  margin = FALSE,
  palette = "default",
  buffer = NULL,
  row = NULL,
  column = NULL,
  block = NULL,
  treatments = NULL,
  ...
)
```

Arguments

`object`	An object to create a plot for. Currently objects from the `multiple_comparisons()` or `design()` functions with class "mct" or "design" respectively are supported.
`...`	Arguments passed to methods.
`size`	Increase or decrease the text size within the plot for treatment labels. Numeric with default value of 4.
`label_height`	Height of the text labels above the upper error bar on the plot. Default is 0.1 (10%) of the difference between upper and lower error bars above the top error bar. Values > 1 are interpreted as the actual value above the upper error bar.
`rotation`	Rotate the x axis labels and the treatment group labels within the plot. Allows for easier reading of long axis or treatment labels. Number between 0 and 360 (inclusive) - default 0
`axis_rotation`	Enables rotation of the x axis independently of the group labels within the plot.
`label_rotation`	Enables rotation of the treatment group labels independently of the x axis labels within the plot.
`margin`	Logical (default `FALSE`). A value of `FALSE` will expand the plot to the edges of the plotting area i.e. remove white space between plot and axes.
`palette`	A string specifying the colour scheme to use for plotting. Default is equivalent to "Spectral". Colour blind friendly palettes can also be provided via options `"colour blind"` (or `"color blind"`, both equivalent to `"viridis"`), `"magma"`, `"inferno"`, `"plasma"` or `"cividis"`. Other palettes from `scales::brewer_pal()` are also possible.
`buffer`	A string specifying the buffer plots to include for plotting. Default is `NULL` (no buffers plotted). Other options are "edge" (outer edge of trial area), "rows" (between rows), "columns" (between columns), "double row" (a buffer row each side of a treatment row) or "double column" (a buffer row each side of a treatment column). "blocks" (a buffer around each treatment block) will be implemented in a future release.

`row`	A variable to plot a column from `object` as rows.
`column`	A variable to plot a column from `object` as columns.
`block`	A variable to plot a column from `object` as blocks.
`treatments`	A variable to plot a column from `object` as treatments.

Value

A `ggplot2` object.

See Also

`multiple_comparisons()` and `design()`

Examples

```
dat.aov <- aov(Petal.Width ~ Species, data = iris)
output <- multiple_comparisons(dat.aov, classify = "Species")
autoplot(output, label_height = 0.5)
des.out <- design(type = "crd", treatments = c(1, 5, 10, 20),
                  reps = 5, nrows = 4, ncols = 5, seed = 42, plot = FALSE)
autoplot(des.out)

# Colour blind friendly colours
autoplot(des.out, palette = "colour-blind")

# Alternative colour scheme
autoplot(des.out, palette = "plasma")
```

---

design — *Create a complete experimental design with graph of design layout and skeletal ANOVA table*

Description

Create a complete experimental design with graph of design layout and skeletal ANOVA table

Usage

```
design(
  type,
  treatments,
  reps,
  nrows,
  ncols,
  brows = NA,
  bcols = NA,
  byrow = TRUE,
  sub_treatments = NULL,
  fac.names = NULL,
  fac.sep = c("", " "),
  plot = TRUE,
  rotation = 0,
  size = 4,
  margin = FALSE,
  save = FALSE,
  savename = paste0(type, "_design"),
  plottype = "pdf",
  seed = TRUE,
  quiet = FALSE,
  ...
)
```

Arguments

`type`	The type of design. Supported design types are `crd`, `rcbd`, `lsd`, `crossed:<type>` where `<type>` is one of the previous types, and `split`. See Details for more information.
`treatments`	A vector containing the treatment names or labels.
`reps`	The number of replicates. Ignored for Latin Square Designs.
`nrows`	The number of rows in the design.
`ncols`	The number of columns in the design.
`brows`	For RCBD and Split Plot designs. The number of rows in a block.
`bcols`	For RCBD and Split Plot designs. The number of columns in a block.
`byrow`	For split-plot only. Logical (default `TRUE`). Provides a way to arrange plots within whole-plots when there are multiple possible arrangements.
`sub_treatments`	A vector of treatments for sub-plots in a split plot design.
`fac.names`	Allows renaming of the `A` level of factorial designs (i.e. those using `agricolae::design.ab()`) by passing (optionally named) vectors of new labels to be applied to the factors within a list. See examples and details for more information.
`fac.sep`	The separator used by `fac.names`. Used to combine factorial design levels. If a vector of 2 levels is supplied, the first separates factor levels and label, and the second separates the different factors.
`plot`	Logical (default `TRUE`). If `TRUE`, display a plot of the generated design. A plot can always be produced later using `autoplot()`.
`rotation`	Rotate the text output as Treatments within the plot. Allows for easier reading of long treatment labels. Takes positive and negative values being number of degrees of rotation from horizontal.
`size`	Increase or decrease the text size within the plot for treatment labels. Numeric with default value of 4.
`margin`	Logical (default `FALSE`). Expand the plot to the edges of the plotting area i.e. remove white space between plot and axes.
`save`	One of `FALSE` (default)/`"none"`, `TRUE`/`"both"`, `"plot"` or `"workbook"`. Specifies which output to save.

`savename`	A file name for the design to be saved to. Default is the type of the design combined with "_design".
`plottype`	The type of file to save the plot as. Usually one of `"pdf"`, `"png"`, or `"jpg"`. See `ggplot2::ggsave()` for all possible options.
`seed`	Logical (default `TRUE`). If `TRUE`, return the seed used to generate the design. If a numeric value, use that value as the seed for the design.
`quiet`	Logical (default `FALSE`). Hide the output.
`...`	Additional parameters passed to `ggplot2::ggsave()` for saving the plot.

Details

The designs currently supported by `type` are Completely Randomised designs (`crd`), Randomised Complete Block designs (`rcbd`), Latin Square Designs (`lsd`), Factorial with crossed structure (use `crossed:<type>` where `<type>` is one of the previous types e.g. `crossed:crd`) and Split Plot designs (`split`). Nested factorial designs are supported through manual setup, see Examples.

If `save = TRUE` (or `"both"`), both the plot and the workbook will be saved to the current working directory, with filename given by `savename`. If one of either `"plot"` or `"workbook"` is specified, only that output is saved. If `save = FALSE` (the default, or equivalently `"none"`), nothing will be output.

`fac.names` can be supplied to provide more intuitive names for factors and their levels in factorial and split plot designs. They can be specified in a list format, for example `fac.names = list(A_names = c("a", "b", "c"), B_names = c("x", "y", "z"))`. This will result a design output with a column named `A_names` with levels `a, b, c` and another named `B_names` with levels `x, y, z`. Labels can also be supplied as a character vector (e.g. `c("A", "B")`) which will result in only the treatment column names being renamed. Only the first two elements of the list will be used, except in the case of a 3-way factorial design.

`...` allows extra arguments to be passed to `ggsave()` for output of the plot. The details of possible arguments can be found in `ggplot2::ggsave()`.

Value

A list containing a data frame with the complete design (`$design`), a ggplot object with plot layout (`$plot.des`), the seed (`$seed`, if `return.seed = TRUE`), and the `satab` object (`$satab`), allowing repeat output of the `satab` table via `cat(output$satab)`.

Examples

```
# Completely Randomised Design
des.out <- design(type = "crd", treatments = c(1, 5, 10, 20),
                  reps = 5, nrows = 4, ncols = 5, seed = 42)

# Randomised Complete Block Design
des.out <- design("rcbd", treatments = LETTERS[1:11], reps = 4,
                  nrows = 11, ncols = 4, brows = 11, bcols = 1, seed = 42)

# Latin Square Design
# Doesn't require reps argument
des.out <- design(type = "lsd", c("S1", "S2", "S3", "S4"),
                  nrows = 4, ncols = 4, seed = 42)

# Factorial Design (Crossed, Completely Randomised)
des.out <- design(type = "crossed:crd", treatments = c(3, 2),
                  reps = 3, nrows = 6, ncols = 3, seed = 42)

# Factorial Design (Crossed, Completely Randomised), renaming factors
des.out <- design(type = "crossed:crd", treatments = c(3, 2),
                  reps = 3, nrows = 6, ncols = 3, seed = 42,
                  fac.names = list(N = c(50, 100, 150),
                                   Water = c("Irrigated", "Rain-fed")))

# Factorial Design (Crossed, Randomised Complete Block Design),
# changing separation between factors
des.out <- design(type = "crossed:rcbd", treatments = c(3, 2),
                  reps = 3, nrows = 6, ncols = 3,
                  brows = 6, bcols = 1,
                  seed = 42, fac.sep = c(":", "_"))

# Factorial Design (Nested, Latin Square)
trt <- c("A1", "A2", "A3", "A4", "B1", "B2", "B3")
des.out <- design(type = "lsd", treatments = trt,
                  nrows = 7, ncols = 7, seed = 42)

# Split plot design
des.out <- design(type = "split", treatments = c("A", "B"), sub_treatments = 1:4,
                  reps = 4, nrows = 8, ncols = 4, brows = 4, bcols = 2, seed = 42)

# Alternative arrangement of the same design as above
des.out <- design(type = "split", treatments = c("A", "B"), sub_treatments = 1:4,
                  reps = 4, nrows = 8, ncols = 4, brows = 4, bcols = 2,
                  byrow = FALSE, seed = 42)
```

des_info *Produce a graph of design layout, skeletal ANOVA table and data frame with complete design*

Description

Produce a graph of design layout, skeletal ANOVA table and data frame with complete design

Usage

```
des_info(
  design.obj,
  nrows,
  ncols,
  brows = NA,
  bcols = NA,
  byrow = TRUE,
  fac.names = NULL,
  fac.sep = c("", " "),
  plot = TRUE,
  rotation = 0,
  size = 4,
  margin = FALSE,
  save = FALSE,
  savename = paste0(design.obj$parameters$design, "_design"),
  plottype = "pdf",
  return.seed = TRUE,
  quiet = FALSE,
  ...
)
```

Arguments

`design.obj`	An `agricolae` design object.
`nrows`	The number of rows in the design.
`ncols`	The number of columns in the design.
`brows`	For RCBD only. The number of rows in a block.
`bcols`	For RCBD only. The number of columns in a block.
`byrow`	For split-plot only. Logical (default: `TRUE`). Provides a way to arrange plots within whole-plots when there are multiple possible arrangements.
`fac.names`	Allows renaming of the `A` level of factorial designs (i.e. those using `agricolae::design.ab()`) by passing (optionally named) vectors of new labels to be applied to the factors within a list. See examples and details for more information.
`fac.sep`	The separator used by `fac.names`. Used to combine factorial design levels. If a vector of 2 levels is supplied, the first separates factor levels and label, and the second separates the different factors.
`plot`	Logical (default `TRUE`). If `TRUE`, display a plot of the generated design. A plot can always be produced later using `autoplot()`.
`rotation`	Rotate the text output as Treatments within the plot. Allows for easier reading of long treatment labels. Takes positive and negative values being number of degrees of rotation from horizontal.
`size`	Increase or decrease the text size within the plot for treatment labels. Numeric with default value of 4.
`margin`	Logical (default FALSE). Setting to `TRUE` will add a margin (white space) between plot and axes.
`save`	One of `FALSE` (default)/`"none"`, `TRUE`/`"both"`, `"plot"` or `"workbook"`. Specifies which output to save.
`savename`	A filename for the design to be saved to. Default is the type of the design combined with "_design".
`plottype`	The type of file to save the plot as. Usually one of `"pdf"`, `"png"`, or `"jpg"`. See `ggplot2::ggsave()` for all possible options.

`return.seed`	Logical (default TRUE). Output the seed used in the design?
`quiet`	Logical (default FALSE). Return the objects without printing output.
`...`	Additional parameters passed to `ggplot2::ggsave()` for saving the plot.

Details

If `save = TRUE` (or `"both"`), both the plot and the workbook will be saved to the current working directory, with filename given by `savename`. If one of either `"plot"` or `"workbook"` is specified, only that output is saved. If `save = FALSE` (the default, or equivalently `"none"`), nothing will be output.

`fac.names` can be supplied to provide more intuitive names for factors and their levels in factorial designs. They should be specified in a list format, for example `fac.names = list(A_names = c("a", "b", "c"), B_names = c("x", "y", "z"))`. This will result a design output with a column named `A_names` with levels `a, b, c` and another named `B_names` with levels `x, y, z`. Only the first two elements of the list will be used.

If `fac.sep` is a single element (e.g. "*"), this is used to separate all factor labels (e.g. A_1_B_1). If it is two elements (e.g. c("", "*")), the first element separates the factor names and their levels, and the second level separates the two factors (e.g. A1_B1).

`...` allows extra arguments to be passed to ggsave for output of the plot. The details of possible arguments can be found in `ggplot2::ggsave()`.

Value

A list containing a data frame with the complete design, a ggplot object with plot layout, the seed (if `return.seed = TRUE`), and the `satab` object, allowing repeat output of the `satab` table via `cat(output$satab)`.

Examples

```
library(agricolae)

# Completely Randomised Design
trt <- c(1, 5, 10, 20)
rep <- 5
outdesign <- design.crd(trt = trt, r = rep, seed = 42)
des.out <- des_info(design.obj = outdesign, nrows = 4, ncols = 5)

# Randomised Complete Block Design
trt <- LETTERS[1:11]
rep <- 4
outdesign <- design.rcbd(trt = trt, r = rep, seed = 42)
des.out <- des_info(
  design.obj = outdesign, nrows = 11,
  ncols = 4, brows = 11, bcols = 1
)

# Latin Square Design
trt <- c("S1", "S2", "S3", "S4")
outdesign <- design.lsd(trt)
```

```
des.out <- des_info(design.obj = outdesign, nrows = 4, ncols = 4)

# Factorial Design (Crossed, Completely Randomised)
trt <- c(3, 2) # Factorial 3 x 2
rep <- 3
outdesign <- design.ab(trt, r = rep, design = "crd")
des.out <- des_info(design.obj = outdesign, nrows = 6, ncols = 3)

# Factorial Design (Crossed, Completely Randomised), renaming factors
trt <- c(3, 2) # Factorial 3 x 2
rep <- 3
outdesign <- design.ab(trt, r = rep, design = "crd")
des.out <- des_info(design.obj = outdesign, nrows = 6, ncols = 3,
                    fac.names = list(N = c(50, 100, 150),
                                     Water = c("Irrigated", "Rain-fed")))

# Factorial Design (Nested, Latin Square)
trt <- c("A1", "A2", "A3", "A4", "B1", "B2", "B3")
outdesign <- design.lsd(trt)
des.out <- des_info(design.obj = outdesign, nrows = 7, ncols = 7)

# Split plot design
trt1 <- c("A", "B")
trt2 <- 1:4
rep <- 4
outdesign <- design.split(trt1, trt2, r = rep)
des.out <- des_info(design.obj = outdesign, nrows = 8, ncols = 4, brows = 4, bcols = 2)
```

---

heat_map — *Produce a heatmap of variables in a grid layout.*

Description

This function plots a heatmap of variables in a grid layout, optionally grouping them.

Usage

```
heat_map(
  data,
  value,
  x_axis,
  y_axis,
  grouping = NULL,
  raster = TRUE,
  smooth = FALSE,
  palette = "default",
  ...
)
```

Arguments

`data`	A data frame containing the data to be plotted.
`value`	A column of `data`, containing the values that vary over the space which produces the colours.
`x_axis`	The column of `data` to use as the x axis data.
`y_axis`	The column of `data` to use as the y axis data.
`grouping`	An optional grouping variable to facet the plot by.
`raster`	Logical (default: `TRUE`). If `TRUE` uses `ggplot2::geom_raster()` for speed. Will not work if the grid is irregular.
`smooth`	Logical (default: `FALSE`). If `raster` is `TRUE`, interpolation can be applied across the grid to obtain a smoothed grid. Ignored if `raster` is `FALSE`.
`palette`	Colour palette to use. By default it will use the `viridis` (colour-blind friendly) palette. Other palettes available can be seen with `grDevices::hcl.pals()`.
`...`	Other arguments passed to `facet_wrap()`

Value

A `ggplot2` object.

Examples

```
set.seed(42)
dat <- expand.grid(x = 1:5, y = 1:6)
dat$value <- rnorm(30)
dat$groups <- sample(rep(LETTERS[1:6], times = 5))

heat_map(dat, value, x, y)

# Column names can be quoted, but don't need to be.
heat_map(dat, "value", "x", "y", "groups")

# Different palettes are available
heat_map(dat, value, x, y, palette = "Spectral")

# Arguments in ... are passed through to facet_wrap
heat_map(dat, value, x, y, groups, labeller = ggplot2:::label_both)
heat_map(dat, value, x, y, groups, scales = "free_y")
heat_map(dat, value, x, y, groups, nrow = 1)
```

`install_asreml` *Install or update the ASReml-R package*

Description

Helper functions for installing or updating the ASReml-R package, intended to reduce the difficulty of finding the correct version for your operating system and R version.

Usage

```
install_asreml(
  library = .libPaths()[1],
  quiet = FALSE,
  force = FALSE,
  keep_file = FALSE
)

update_asreml(...)
```

Arguments

`library`	Library location to install ASReml-R. Uses first option in `.libPaths()` by default.
`quiet`	Logical (default `FALSE`). Should package be installed quietly?
`force`	Logical (default `FALSE`). Force ASReml-R to install. Useful for upgrading if it is already installed.
`keep_file`	Should the downloaded asreml package file be kept? Default is `FALSE`. `TRUE` downloads to current directory. A file path can also be provided to save to another directory. See `Details` for more information.
`...`	other arguments passed to `install_asreml()`

Details

The ASReml-R package file is downloaded from a shortlink, and if `keep_file` is `TRUE`, the package archive file will be saved in the current directory. If a valid path is provided in `keep_file`, the file will be saved to that path, but all directories are assumed to exist and will not be created. If `keep_file` does not specify an existing, valid path, an error will be shown after package installation.

Value

Silently returns `TRUE` if `asreml` installed successfully or already present, `FALSE` otherwise. Optionally prints a confirmation message on success.

Examples

```
## Not run:
# Example 1: download and install asreml
install_asreml()

# Example 2: install asreml and save file for later
install_asreml(keep_file = TRUE)

## End(Not run)
```

`logl_test` *Conduct a log-likelihood test for comparing terms in ASReml-R models*

Description

Conduct a log-likelihood test for comparing terms in ASReml-R models

Usage

```
logl_test(
  model.obj,
  rand.terms = NULL,
  resid.terms = NULL,
  decimals = 3,
  numeric = FALSE,
  quiet = FALSE
)
```

Arguments

`model.obj`	An ASReml-R model object
`rand.terms`	Random terms from the model. Default is NULL.
`resid.terms`	Residual terms from the model. Default is NULL.
`decimals`	Controls rounding of decimal places in output. Default is 3 decimal places.
`numeric`	Return p-values as numeric? Default is that they are characters, where very small values shown as less than a small number. See `details` for more.
`quiet`	Logical (default: `FALSE`). Hide warnings and messages?

Details

Typically p-values cannot be 0, and are usually just below some threshold of accuracy in calculation of probability.

Value

A dataframe containing the results of the test.

Examples

```
## Not run:
library(asreml)
dat <- asreml::oats
dat <- dat[order(dat$Row, dat$Column),]

#Fit ASReml Model
model.asr <- asreml(yield ~ Nitrogen + Variety + Nitrogen:Variety,
                    random = ~ Blocks + Blocks:Wplots,
                    residual = ~ ar1(Row):ar1(Column),
                    data = dat)
oats.logl <- logl_test(
  model.obj = model.asr, rand.terms = c("Blocks", "Blocks:Wplots"),
  resid.terms = c("ar1(Row)", "ar1(Column)")
)
oats.logl

## End(Not run)
```

multiple_comparisons *Perform Multiple Comparison Tests on a statistical model*

Description

A function for comparing and ranking predicted means with Tukey's Honest Significant Difference (HSD) Test.

Usage

```
multiple_comparisons(
  model.obj,
  classify,
  sig = 0.05,
  int.type = "ci",
  trans = NA,
  offset = NA,
  power = NA,
  decimals = 2,
  descending = FALSE,
  plot = FALSE,
  label_height = 0.1,
  rotation = 0,
  save = FALSE,
  savename = "predicted_values",
  order,
  pred.obj,
  pred,
  ...
)
```

Arguments

`model.obj`	An ASReml-R or aov model object. Will likely also work with `lme` (`nlme::lme()`), `lmerMod` (`lme4::lmer()`) models as well.
`classify`	Name of predictor variable as string.

`sig`	The significance level, numeric between 0 and 1. Default is 0.05.
`int.type`	The type of confidence interval to calculate. One of `ci`, `1se` or `2se`. Default is `ci`.
`trans`	Transformation that was applied to the response variable. One of `log`, `sqrt`, `logit`, `power` or `inverse`. Default is `NA`.
`offset`	Numeric offset applied to response variable prior to transformation. Default is `NA`. Use 0 if no offset was applied to the transformed data. See Details for more information.
`power`	Numeric power applied to response variable with power transformation. Default is `NA`. See Details for more information.
`decimals`	Controls rounding of decimal places in output. Default is 2 decimal places.
`descending`	Logical (default `FALSE`). Order of the output sorted by the predicted value. If `TRUE`, largest will be first, through to smallest last.
`plot`	Automatically produce a plot of the output of the multiple comparison test? Default is `FALSE`. This is maintained for backwards compatibility, but the preferred method now is to use `autoplot(<multiple_comparisons output>)`. See `autoplot.mct()` for more details.
`label_height`	Height of the text labels above the upper error bar on the plot. Default is 0.1 (10%) of the difference between upper and lower error bars above the top error bar.
`rotation`	Rotate the text output as Treatments within the plot. Allows for easier reading of long treatment labels. Number between 0 and 360 (inclusive) - default 0
`save`	Logical (default `FALSE`). Save the predicted values to a csv file?
`savename`	A file name for the predicted values to be saved to. Default is `predicted_values`.
`order`	Deprecated. Use `descending` instead.
`pred.obj`	Deprecated. Predicted values are calculated within the function from version 1.0.1 onwards.
`pred`	Deprecated. Use `classify` instead.
`...`	Other arguments passed through to `predict.asreml()`.

Details

Some transformations require that data has a small offset applied, otherwise it will cause errors (for example taking a log of 0, or square root of negative values). In order to correctly reverse this offset, if the `trans` argument is supplied, an offset value must also be supplied. If there was no offset required for a transformation, then use a value of 0 for the `offset` argument.

Value

A list containing a data frame with predicted means, standard errors, confidence interval upper and lower bounds, and significant group allocations (named `predicted_values`), as well as a plot visually displaying the predicted values (named `predicted_plot`). If some of the predicted values are aliased, a warning is printed, and the aliased treatment levels are returned in the output (named `aliased`).

References

Jørgensen, E. & Pedersen, A. R. (1997). How to Obtain Those Nasty Standard Errors From Transformed Data - and Why They Should Not Be Used. https://pure.au.dk/portal/en/publications/how-to-obtain-those-nasty-standard-errors-from-transformed-data--and-why-they-should-
.html

Examples

```
# Fit aov model
model <- aov(Petal.Length ~ Species, data = iris)

# Display the ANOVA table for the model
anova(model)

# Determine ranking and groups according to Tukey's Test
pred.out <- multiple_comparisons(model, classify = "Species")

# Display the predicted values table
pred.out

# Show the predicted values plot
autoplot(pred.out, label_height = 0.5)



## Not run:
# ASReml-R Example
library(asreml)

#Fit ASReml Model
model.asr <- asreml(yield ~ Nitrogen + Variety + Nitrogen:Variety,
                    random = ~ Blocks + Blocks:Wplots,
                    residual = ~ units,
                    data = asreml::oats)

wald(model.asr) #Nitrogen main effect significant

#Determine ranking and groups according to Tukey's Test
pred.out <- multiple_comparisons(model.obj = model.asr, classify = "Nitrogen",
                    descending = TRUE, decimals = 5)

pred.out

# Example using a box-cox transformation
set.seed(42) # See the seed for reproducibility
resp <- rnorm(n = 50, 5, 1)^3
trt <- as.factor(sample(rep(LETTERS[1:10], 5), 50))
block <- as.factor(rep(1:5, each = 10))
ex_data <- data.frame(resp, trt, block)

# Change one treatment random values to get significant difference
ex_data$resp[ex_data$trt=="A"] <- rnorm(n = 5, 7, 1)^3
```

```
model.asr <- asreml(resp ~ trt,
                    random = ~ block,
                    residual = ~ units,
                    data = ex_data)

resplot(model.asr)

# Perform Box-Cox transformation and get maximum value
out <- MASS::boxcox(ex_data$resp~ex_data$trt)
out$x[which.max(out$y)] # 0.3838

# Fit cube root to the data
model.asr <- asreml(resp^(1/3) ~ trt,
                    random = ~ block,
                    residual = ~ units,
                    data = ex_data)
resplot(model.asr) # residual plots look much better

#Determine ranking and groups according to Tukey's Test
pred.out <- multiple_comparisons(model.obj = model.asr,
                                 classify = "trt",
                                 trans = "power", power = (1/3))

pred.out
autoplot(pred.out)

## End(Not run)
```

print.mct *Print output of multiple_comparisons*

Description

Print output of multiple_comparisons

Usage

```
## S3 method for class 'mct'
print(x, ...)
```

Arguments

`x`	An mct object to print to the console.
`...`	Other arguments

Value

The original object invisibly.

See Also

multiple_comparisons()

Examples

```
dat.aov <- aov(Petal.Width ~ Species, data = iris)
output <- multiple_comparisons(dat.aov, classify = "Species")
print(output)
```

---

resplot *Produce residual plots of linear models*

Description

Produces plots of residuals for assumption checking of linear (mixed) models.

Usage

```
resplot(
  model.obj,
  shapiro = TRUE,
  call = FALSE,
  label.size = 10,
  axes.size = 10,
  call.size = 9,
  mod.obj
)
```

Arguments

model.obj	An aov, lm, lme (nlme::lme()), lmerMod (lme4::lmer()), asreml or mmer (sommer) model object.
shapiro	(Logical) Display the Shapiro-Wilks test of normality on the plot?
call	(Logical) Display the model call on the plot?
label.size	A numeric value for the size of the label (A,B,C) font point size.
axes.size	A numeric value for the size of the axes label font size in points.
call.size	A numeric value for the size of the model displayed on the plot.
mod.obj	Deprecated to be consistent with other functions. Please use model.obj instead.

Value

A ggplot2 object containing the diagnostic plots.

Examples

```
dat.aov <- aov(Petal.Length ~ Petal.Width, data = iris)
resplot(dat.aov)
resplot(dat.aov, call = TRUE)
```

`summary_graph` *Visualise a graphical summary of variables from a data frame*

Description

Variables are plotted in different ways according to the number of explanatory variables provided as input.

Usage

```
summary_graph(data, response, exp_var, resp_units = "")
```

Arguments

`data`	A data frame containing the variables to be plotted.
`response`	The response variable to plot.
`exp_var`	The explanatory (or grouping) variable(s) to plot. Up to three can be provided.
`resp_units`	A string providing units to display on the response variable (y) axis. Will use the empty string by default so axes will have no units by default.

Details

With a single explanatory variable, a boxplot grouped by `exp_var` is produced. With two explanatory variables, a dot-plot with lines connecting the mean of each group is produced, with the first element of `exp_var` used as the x axis variable, and the second is used to colour the points. Three explanatory variables produces the same as two, but with the third used to facet the plot.

Value

A ggplot2 plot object

Examples

```
summary_graph(iris, "Petal.Length", "Species", "mm")

# Multiple
summary_graph(npk, "yield", c("N", "P"), "lb/plot")

summary_graph(npk, "yield", c("N", "P", "K"), "lb/plot")
```

`variogram` *Display variogram plots for spatial models*

Description

Produces variogram plots for checking spatial trends.

Usage

```
variogram(
  model.obj,
  row = NA,
  column = NA,
  horizontal = TRUE,
  palette = "default"
)
```

Arguments

`model.obj`	An `asreml` model object.
`row`	A row variable.
`column`	A column variable.
`horizontal`	Logical (default `TRUE`). The direction the plots are arranged. The default `TRUE` places the plots above and below, while `FALSE` will place them side by side.
`palette`	A string specifying the colour scheme to use for plotting. The default value (`"default"`) is equivalent to `"rainbow"`. Colour blind friendly palettes can also be provided via options `"colo(u)r blind"` (both equivalent to `"viridis"`), `"magma"`, `"inferno"`, `"plasma"`, `"cividis"`, `"rocket"`, `"mako"` or `"turbo"`. The `"Spectral"` palette from `scales::brewer_pal()` is also possible.

Value

A `ggplot2` object.

References

S. P. Kaluzny, S. C. Vega, T. P. Cardoso, A. A. Shelly, "S+SpatialStats: User's Manual for Windows® and UNIX®" *Springer New York*, 2013, p. 68, https://books.google.com.au/books?id=iADkBwvario_pointsQBAJ.

A. R. Gilmour, B. R. Cullis, A. P. Verbyla, "Accounting for Natural and Extraneous Variation in the Analysis of Field Experiments." *Journal of Agricultural, Biological, and Environmental Statistics 2, no. 3*, 1997, pp. 269–93, https://doi.org/10.2307/1400446.

Examples

```
## Not run:
library(asreml)
oats <- asreml::oats
oats <- oats[order(oats$Row, oats$Column),]
model.asr <- asreml(yield ~ Nitrogen + Variety + Nitrogen:Variety,
                    random = ~ Blocks + Blocks:Wplots,
                    residual = ~ ar1(Row):ar1(Column),
                    data = oats)
variogram(model.asr)

## End(Not run)
```

Index