Package 'biometryassist'

May 7, 2026

Type Package

Title Functions to Assist Design and Analysis of Agronomic Experiments

Version 1.4.0

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 (>= 4.1.0)

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

Suggests covr, crayon, ggspatial, knitr, Matrix, mockery, openxlsx2, rmarkdown, testthat, vdiffr, withr

Enhances asreml, ARTool, lme4, nlme, sommer

Config/testthat/edition 3

Config/testthat/parallel true

Encoding UTF-8

RoxygenNote 7.3.3

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 2026-02-03 11:30:02 UTC

Contents

`add_buffers` *Add buffers to an existing design*

Description

Add buffers to an existing design

Usage

```
add_buffers(design_obj, type)
```

Arguments

`design_obj`	A design object (with class "design") from the design() function
`type`	The type of buffer to add. One of 'edge', 'row', 'column', 'double row', or 'double column'.

Value

The modified design object with buffers added

Examples

```
# Create a simple CRD design
des <- design(type = "crd", treatments = c("A", "B"), reps = 3, nrows = 2, ncols = 3, seed = 42)

# Plot the original design
autoplot(des)

# Add edge buffers to the design
des_buf <- add_buffers(des, type = "edge")

# Plot the design with buffers
autoplot(des_buf)

# Add double row buffers
des_row_buf <- add_buffers(des, type = "double row")
autoplot(des_row_buf)
```

---

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,
  type = "point",
  ...
)

## S3 method for class 'design'
autoplot(
  object,
```

```
  rotation = 0,
  size = 4,
  margin = FALSE,
  palette = "default",
  row = NULL,
  column = NULL,
  block = NULL,
  treatments = NULL,
  legend = TRUE,
  ...
)
```

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.
type	A string specifying the type of plot to display. The default of 'point' will display a point estimate with error bars. The alternative, 'column' (or 'col'), will display a column graph with error bars.
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 or a vector of custom colours to use as the palette. Default is equivalent to "Spectral". Colour blind friendly palettes can also be provided via options "colour blind" (or "colour blind", both equivalent to "viridis"), "magma", "inferno", "plasma", "cividis", "rocket", "mako" or "turbo". Other palettes from scales::brewer_pal() are also possible.
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.
legend	Logical (default TRUE). If TRUE, displays the legend for treatment colours.

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")

# Custom colour palette
autoplot(des.out, palette = c("#ef746a", "#3fbfc5", "#81ae00", "#c37cff"))

# Visualise different components of a 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)

# Show the wholeplot components
autoplot(des.out, treatments = wholeplots)

# Display block level
autoplot(des.out, treatments = block)
```

---

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("", " "),
  buffer = NULL,
  plot = TRUE,
  rotation = 0,
  size = 4,
  margin = FALSE,
  save = FALSE,
  savename = paste0(type, "_design"),
  plottype = "pdf",
  seed = TRUE,
  quiet = FALSE,
  ...
)
```

Arguments

type	The design type. One of crd, rcbd, lsd, split, strip, or a crossed factorial specified as crossed:<base> where <base> is one of crd, rcbd, or lsd.
treatments	A vector containing the treatment names or labels. For split-plot designs, these treatments are applied to whole-plots. For strip-plot designs, these treatments are applied to row-strips (entire rows within each block receive the same treatment).
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, split-plot and strip-plot designs. The number of rows in a block.
bcols	For RCBD, split-plot and strip-plot designs. The number of columns in a block.
byrow	For split-plot and strip-plot designs. Logical (default TRUE). Controls the within-block arrangement when there are multiple valid layouts.
sub_treatments	A vector of treatments for the sub-plot factor (required for split and strip). For strip-plot designs, these treatments are applied to column-strips (entire columns within each block receive the same treatment). To apply treatments to columns instead of rows, swap the treatments and sub_treatments arguments.
fac.names	Allows renaming of the A level of factorial designs 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.

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.
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

Supported designs are Completely Randomised (crd), Randomised Complete Block (rcbd), Latin Square (lsd), split-plot (split), strip-plot (strip), and crossed factorial designs via crossed:<base> where <base> is crd, rcbd, or lsd (e.g. crossed:crd).

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)

# Strip plot design
```

```
des.out <- design(type = "strip", treatments = c("A", "B", "C"), sub_treatments = 1:4,
                  reps = 4, nrows = 12, ncols = 4, brows = 3, bcols = 4, seed = 42)
```

`export_design_to_excel` *Export Experimental Design Layout to Excel*

Description

Converts an experimental design dataframe into a spatial layout matrix and exports to Excel with optional colour coding by treatment.

Usage

```
export_design_to_excel(
  design_df,
  value_column = "treatments",
  filename = "experimental_design.xlsx",
  palette = "default"
)
```

Arguments

`design_df`	A dataframe containing experimental design with 'row' and 'col' columns
`value_column`	Character string specifying which column to use for layout values (default: "treatments")
`filename`	Character string for Excel filename (default: "experimental_design.xlsx")
`palette`	colour palette for treatments. Can be a palette name (see details) or vector of colours. Set to NULL to disable colouring (default: "default")

Details

This function takes an experimental design in long format (with row/col coordinates) and converts it to a matrix layout that matches the spatial arrangement of the experiment, then exports to Excel with formatting and optional colour coding.

Valid palette options include:

- "default" - Spectral palette
- ColorBrewer palettes: "brbg", "piyg", "prgn", "puor", "rdbu", "rdgy", "rdylbu", "rdylgn", "spectral", "set3", "paired"
- Viridis palettes: "viridis", "magma", "inferno", "cividis", "plasma", "rocket", "mako", "turbo"
- colour blind friendly: "colour blind", "color blind", "cb" (uses viridis)
- Custom vector of colours (must match number of unique treatments)

Requires the 'openxlsx2' package to be installed.

Value

Invisibly returns the layout dataframe

Examples

```
## Not run:
# Export with default colours
export_design_to_excel(my_design, "treatments", "my_design.xlsx")

# Export without colours
export_design_to_excel(my_design, "treatments", "my_design.xlsx", palette = NULL)

# Export with custom palette
export_design_to_excel(my_design, "treatments", "my_design.xlsx", palette = "viridis")

## End(Not run)
```

---

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 ggplot2::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,
  check_version = TRUE
)

update_asreml(...)
```

Arguments

library	Library location to install ASReml-R. Uses first option in .libPaths() by default.
quiet	Logical or character (default FALSE). Controls output verbosity. FALSE shows normal messages, TRUE suppresses messages, "verbose" shows detailed debugging information.
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.
check_version	Logical (default TRUE). Should function check if there is a newer version of asreml available before attempting to download and install?
...	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)

# Example 3: install with verbose debugging
install_asreml(quiet = "verbose")

## End(Not run)
```

list_templates *List available biometryassist templates*

Description

list_templates() returns a character vector of available analysis templates included with the biometryassist package.

Usage

```
list_templates()
```

Value

character() Vector of available template file names.

See Also

use_template() to copy and use a template

Examples

```
# See what templates are available
list_templates()
```

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	Character vector of random terms to test. Default is NULL.
resid.terms	Character vector of residual terms to test. Default is NULL.
decimals	Number of decimal places to round p-values. Default is 3.
numeric	Logical. Should p-values be returned as numeric? Default is FALSE (formatted).
quiet	Logical. Suppress model update messages and warnings? Default is FALSE.

Value

A data frame of terms and corresponding log-likelihood ratio test p-values.

---

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 = NULL,
  offset = NULL,
  power = NULL,
  decimals = 2,
  descending = FALSE,
  groups = TRUE,
  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, tukey, 1se, 2se, or none. Default is ci.
trans	Transformation that was applied to the response variable. One of log, sqrt, logit, power, inverse, or arcsin. Default is NULL.
offset	Numeric offset applied to response variable prior to transformation. Default is NULL. 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 NULL. 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.
groups	Logical (default TRUE). If TRUE, the significance letter groupings will be calculated and displayed. This can get overwhelming for large numbers of comparisons, so can be turned off by setting to FALSE.
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 get_predictions().

Details

Offset:

Some transformations require that data has a small offset to be applied, otherwise it will cause errors (for example taking a log of 0, or the square root of negative values). In order to correctly

reverse this offset, if the `trans` argument is supplied, a value should also be supplied in the `offset` argument. By default the function assumes no offset was required for a transformation, implying a value of 0 for the `offset` argument. If an offset value is provided, use the same value as provided in the model, not the inverse. For example, if adding 0.1 to values for a log transformation, add 0.1 in the `offset` argument.

Power:

The power argument allows the specification of arbitrary powers to be back transformed, if they have been used to attempt to improve normality of residuals.

Confidence Intervals & Comparison Intervals:

The function provides several options for confidence intervals via the `int.type` argument:

- `ci` **(default)**: Traditional confidence intervals for individual means. These estimate the precision of each individual mean but may not align with the multiple comparison results. Non-overlapping traditional confidence intervals do not necessarily indicate significant differences in multiple comparison tests.
- `tukey`: Tukey comparison intervals that are consistent with the multiple comparison test. These intervals are wider than regular confidence intervals and are designed so that non-overlapping intervals correspond to statistically significant differences in the Tukey HSD test. This ensures visual consistency between the intervals and letter groupings.
- `1se` and `2se`: Intervals of ±1 or ±2 standard errors around each mean.
- `none`: No confidence intervals will be calculated or displayed in plots.

By default, the function displays regular confidence intervals (`int.type = "ci"`), which estimate the precision of individual treatment means. However, when performing multiple comparisons, these regular confidence intervals may not align with the letter groupings from Tukey's HSD test. Specifically, you may observe non-overlapping confidence intervals for treatments that share the same letter group (indicating no significant difference).

This occurs because regular confidence intervals and Tukey's HSD test serve different purposes:

- Regular confidence intervals estimate individual mean precision
- Tukey's HSD controls the family-wise error rate across all pairwise comparisons

To resolve this visual inconsistency, you can use Tukey comparison intervals (`int.type = "tukey"`). These intervals are specifically designed for multiple comparisons and will be consistent with the letter groupings: non-overlapping Tukey intervals indicate significant differences, while overlapping intervals suggest no significant difference.

The function will issue a message if it detects potential inconsistencies between non-overlapping confidence intervals and letter groupings, suggesting the use of Tukey intervals for clearer interpretation. For multiple comparison contexts, Tukey comparison intervals are recommended as they provide visual consistency with the statistical test being performed and avoid the common confusion where traditional confidence intervals don't overlap but groups share the same significance letter.

Value

An object of class `mct` (a list with class attributes) containing:

`predictions` A data frame with predicted means, standard errors, confidence interval upper and lower bounds, and significant group allocations

pairwise_pvalues
: A symmetric matrix of p-values for all pairwise comparisons using Tukey's HSD test

hsd
: The Honest Significant Difference value(s) used in the comparisons. Either a single numeric value (if constant across comparisons) or a matrix (if varies by comparison)

aliased
: Character vector of aliased treatment levels (only present if some predictions are aliased)

sig_level
: The significance level used (default 0.05)

References

Jørgensen, E. & Pedersen, A. R. (1997). How to Obtain Those Nasty Standard Errors From Transformed Data - and Why They Should Not Be Used.

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 (with Tukey intervals)
pred.out <- multiple_comparisons(model, classify = "Species")

# Display the predicted values table
pred.out

# Access the p-value matrix
pred.out$pairwise_pvalues

# Access the HSD value
pred.out$hsd

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

# Use traditional confidence intervals instead of Tukey comparison intervals
pred.out.ci <- multiple_comparisons(model, classify = "Species", int.type = "ci")
pred.out.ci

# Plot without confidence intervals
pred.out.none <- multiple_comparisons(model, classify = "Species", int.type = "none")
autoplot(pred.out.none)

# AOV model example with transformation
my_iris <- iris
my_iris$Petal.Length <- exp(my_iris$Petal.Length) # Create exponential response
exp_model <- aov(Petal.Length ~ Species, data = my_iris)
```

```
resplot(exp_model) # Residual plot shows problems

# Fit a new model using a log transformation of the response
log_model <- aov(log(Petal.Length) ~ Species, data = my_iris)

resplot(log_model) # Looks much better

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

# Back transform, because the "original" data was exponential
pred.out <- multiple_comparisons(log_model, classify = "Species",
                                 trans = "log")

# Display the predicted values table
pred.out

# Show the predicted values plot
autoplot(pred.out, label_height = 15)

## 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)

# lambda = 1/3 from MASS::boxcox()
model.asr <- asreml(resp^(1/3) ~ trt,
                    random = ~ block,
                    residual = ~ units,
                    data = ex_data)

resplot(model.asr) # Look much better

pred.out <- multiple_comparisons(model.obj = model.asr,
                                 classify = "trt",
                                 trans = "power", power = (1/3))

pred.out
autoplot(pred.out, label_height = 0.5)

## 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 passed to print.data.frame

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,
  onepage = FALSE,
  onepage_cols = 3,
  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-Wilk test of normality on the plot? This test is unreliable for larger numbers of observations and will not work with n >= 5000 so will be omitted from any plots.
`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.
`onepage`	(Logical) If TRUE and there are multiple plots, combines up to 6 plots per page.
`onepage_cols`	Integer. Number of columns to use in grid layout when onepage=TRUE. Default is 3.
`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 explanatory variables can be provided as a vector
summary_graph(npk, "yield", c("N", "P"), "lb/plot")

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

use_template *Use biometryassist analysis templates*

Description

use_template() copies a pre-built analysis template from the biometryassist package to your working directory and optionally opens it for editing. These templates provide standardized approaches for common agronomic analyses.

Usage

```
use_template(
  template_name = "mixed_model_template.R",
  dest_dir = ".",
  open = TRUE,
  overwrite = FALSE,
  output_name = NULL
)
```

Arguments

template_name	Name or path of the template file to use. Default is "mixed_model_template.R". Available templates can be listed with list_templates().
dest_dir	Directory where the template should be copied. Default is the current working directory ("..").
open	Logical (default TRUE). Should the template file be opened in the default editor after copying?
overwrite	Logical (default FALSE). Should existing files be overwritten?
output_name	character, Optional. Name for the copied file in the destination directory. If not specified, defaults to "analysis_script.R". If specified, the template will be copied and renamed to this file. (The original template file is not overwritten.)

Details

This function is designed to help users get started with biometryassist analyses by providing tested, documented templates. The templates include:

- Suggested package loading
- Commented code explaining steps
- Example data exploration
- Common analysis workflows

If a file with the same name already exists in the destination directory, the function will not overwrite it unless overwrite = TRUE is specified. In this case, it will still open the existing file if open = TRUE.

Value

The file path to the copied template (invisibly). Called primarily for its side effects of copying and optionally opening the template file.

See Also

- `list_templates()` to see available templates

Examples

```
## Not run:
# Copy and open the default analysis template
use_template()

# Copy a specific template without opening
use_template("anova_template.R", open = FALSE)

# Copy to a specific directory
use_template("mixed_model_template.R", dest_dir = "analyses")

# Overwrite an existing file
use_template("mixed_model_template.R", overwrite = TRUE)

## End(Not run)
```

---

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 = "rainbow",
  onepage = FALSE
)
```

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.
onepage	Logical (default FALSE). If TRUE and there are multiple groups, combines up to 6 plots onto a single page using a grid layout.

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