Package 'biometryassist'

October 7, 2026

Type Package

Title Functions to Assist Design and Analysis of Agronomic Experiments

Version 1.5.1

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 (>= 4.1.0)

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

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

Enhances afex, ARTool, asreml, glmmTMB, lme4, lme4breeding, lmerTest, nlme, sommer

Config/testthat/edition 3

Config/testthat/parallel true

Encoding UTF-8

VignetteBuilder quarto

Config/roxygen2/version 8.1.0

NeedsCompilation no

Author Sharon Nielsen [aut],
Sam Rogers [aut, cre],
Annie Conway [aut],
Michael Mumford [ctb],
Adelaide University [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 2026-10-07 16:50:13 UTC

Contents

add_buffers . . . 2
autoplot . . . 4
autoplot.design . . . 4
autoplot.mct . . . 6
autoplot.pairwise_comparisons . . . 8
autoplot.reference_comparisons . . . 13
design . . . 19
export_design_to_excel . . . 22
heat_map . . . 24
install_asreml . . . 25
list_templates . . . 27
logl_test . . . 27
multiple_comparisons . . . 28
print.mct . . . 36
resplot . . . 36
summary_graph . . . 38
use_template . . . 39
variogram . . . 40

Index **43**

`add_buffers` *Add buffers to an existing design*

Description

Add buffers to an existing design

Usage

```
add_buffers(design_obj, type, by = NULL)
```

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', 'double column', 'blocks', or 'double block'/'entire block'/'full block'.
by	The name of a grouping column. Used by the block-based buffer types ("block" and the "double block" family), and optionally by "row"/ "column" to insert a buffer only where the group changes between adjacent rows/columns (e.g. between wholeplots in a split-plot design). Defaults to "block" when the design has a "block" column, otherwise NULL.

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 (a border around the whole field) and plot
autoplot(add_buffers(des, type = "edge"))

# A buffer row between every pair of rows
autoplot(add_buffers(des, type = "row"))

# Two buffer rows between every pair of rows
autoplot(add_buffers(des, type = "double row"))

# For block designs, buffer the boundaries between blocks
des_rcbd <- design(type = "rcbd", treatments = LETTERS[1:4], reps = 4,
                   nrows = 4, ncols = 4, brows = 2, bcols = 2, seed = 42)

# One buffer at each internal block boundary
autoplot(add_buffers(des_rcbd, type = "block"))

# A full buffer ring surrounding each block
autoplot(add_buffers(des_rcbd, type = "double block"))

# Use `by` to buffer where a grouping column changes. In a split-plot the
# `plots` column identifies each wholeplot, so this rings every wholeplot:
des_sp <- design(type = "split", treatments = c("A", "B"), sub_treatments = 1:4,
                 reps = 4, nrows = 8, ncols = 4, brows = 4, bcols = 2, seed = 42)
autoplot(add_buffers(des_sp, type = "double block", by = "plots"))

# ... or just a buffer row between the wholeplot bands
autoplot(add_buffers(des_sp, type = "row", by = "plots"))
```

autoplot *Generate automatic plots for objects generated in biometryassist*

Description

ggplot2::autoplot() methods are provided for the objects created by biometryassist. See the per-class methods for the available options: autoplot.mct() for multiple_comparisons() output, autoplot.design() for design() output, and autoplot.pairwise_comparisons() / autoplot.reference_comparisons(), which are documented alongside their respective functions.

Usage

```
autoplot(object, ...)
```

Arguments

object	An object created by biometryassist. Methods are provided for the mct, design, pairwise_comparisons and reference_comparisons classes.
...	Arguments passed to the individual autoplot methods.

Value

A ggplot2 object.

See Also

autoplot.mct(), autoplot.design(), multiple_comparisons() and design()

autoplot.design *Plot the layout of an experimental design*

Description

Produces a plot of the plot/field layout for a design() result, with plots coloured by treatment and block boundaries drawn for blocked designs.

Usage

```
## 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	A design object, as produced by design().
rotation	Rotate the treatment labels within the plot. Allows for easier reading of long treatment labels. Number between 0 and 360 (inclusive) - default 0
size	Increase or decrease the text size within the plot for treatment labels. Numeric with default value of 4.
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.
...	Arguments passed to ggplot2::geom_text() for the plot labels.

Value

A ggplot2 object.

See Also

design()

Examples

```
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)
```

`autoplot.mct` *Plot the predicted means from a multiple comparisons test*

Description

Produces a plot of the predicted means from a `multiple_comparisons()` result, with error bars (or a single Tukey's HSD reference bar) and significance-group lettering.

Usage

```
## S3 method for class 'mct'
autoplot(
  object,
  size = 4,
  label_height = 0.1,
  rotation = 0,
  axis_rotation = rotation,
  label_rotation = rotation,
  type = "point",
  errorbar_type = "ci",
  include_errorbar = TRUE,
  include_lettering = TRUE,
  trans_scale = FALSE,
  ...
)
```

Arguments

`object`	An `mct` object, as produced by `multiple_comparisons()`.
`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. One of "point" (the default; point estimates), "line" (point estimates joined by a line), or "column" (also "col" or "bar"; a column graph). Error bars are added according to errorbar_type unless include_errorbar = FALSE.

errorbar_type — A string (default "ci") specifying what the error bars represent. "ci" draws an interval around each mean (the interval type chosen via int.type in multiple_comparisons()). "hsd" draws a single Tukey's Honest Significant Difference reference bar instead of per-mean intervals. An HSD bar is only meaningful on the model (transformed) scale, so requesting "hsd" plots the means on that scale.

include_errorbar — Logical (default TRUE). Whether to draw error bars. FALSE omits them entirely (the errorbar_type is then ignored).

include_lettering — Logical (default TRUE). Whether to draw the significance-group lettering above the means.

trans_scale — Logical (default FALSE). When the means were back-transformed in multiple_comparisons(), FALSE plots them on the original (back-transformed) scale, while TRUE plots them on the model (transformed) scale and adds a back-transformed secondary axis. Has no effect when no transformation was used.

... — Arguments passed to ggplot2::element_text() for the axis and label text.

Value

A ggplot2 object.

See Also

multiple_comparisons()

Examples

```
dat.aov <- aov(Petal.Width ~ Species, data = iris)
output <- multiple_comparisons(dat.aov, classify = "Species")
autoplot(output, label_height = 0.5)

# Join the means with a line
autoplot(output, type = "line", label_height = 0.5)
```

```
# Show a single Tukey's HSD reference bar instead of per-mean intervals
autoplot(output, errorbar_type = "hsd")
```

`autoplot.pairwise_comparisons` *Pairwise comparisons of predicted means*

Description

Test a chosen set of pairwise differences between the predicted means of a fitted model, with multiplicity adjustment over the chosen set and optional splitting into independent subgroups. Unlike `multiple_comparisons()`, which is means-centric and summarises *all* pairwise comparisons via a compact letter display, `pairwise_comparisons()` is difference-centric: it returns a tidy table with one row per requested comparison. This honestly represents selective and irregular comparison sets, for which letter groupings are not valid.

Usage

```
## S3 method for class 'pairwise_comparisons'
autoplot(object, ..., axis_rotation = 0, label_rotation = 0)

pairwise_comparisons(
  model.obj,
  classify,
  pairs = NULL,
  contrasts = NULL,
  adjust = "holm",
  by = NULL,
  sig = 0.05,
  include_means = TRUE,
  descending = NULL,
  ...
)

## S3 method for class 'pairwise_comparisons'
print(x, decimals = 2, ...)
```

Arguments

`object`	A `pairwise_comparisons` object.
`...`	Other arguments passed to the model-specific prediction methods (e.g. ASReml-R `predict()` arguments such as `present`; see *ASReml-R prediction arguments* below).
`axis_rotation`	Rotation (degrees) of the x-axis (estimate) labels.
`label_rotation`	Rotation (degrees) of the y-axis (comparison) labels.

model.obj	An asreml, aov, lm, lme (nlme::lme()) or lmerMod (lme4::lmer()) model object.
classify	Name of the predictor variable(s) to compare, as a string. Interactions are specified with : (e.g. "Trt:Site"). For asreml models, see *ASReml-R terms in* classify below.
pairs	The comparisons to test. NULL (default) tests all pairwise comparisons. Otherwise either a character vector of "level1-level2" labels (levels of an interaction joined by :, the two sides of a pair separated by -, e.g. "A:X-B:Y"), or a list of length-2 character vectors (e.g. list(c("A:X", "B:Y"))). Level names may contain -; the list form is only needed if a pair is genuinely ambiguous. See Details.
contrasts	An optional named list of general linear contrasts to test instead of pairs. Each element is a named numeric vector of coefficients keyed by level label (e.g. list("A vs B & C" = c(A = 1, B = -0.5, C = -0.5))), and the list names become the comparison labels. The estimate is the corresponding linear combination of the predicted means. Mutually exclusive with pairs; coefficients should sum to zero (a warning is issued otherwise). include_means does not apply to this form. See Details for how the standard error and degrees of freedom are obtained.
adjust	The method used to adjust p-values for multiplicity over the chosen set, passed to stats::p.adjust(). Default is "holm". Any stats::p.adjust.methods value is accepted. "tukey" is **not** valid here (it is exact only for the complete set of all pairwise comparisons — use multiple_comparisons() for that).
by	A character vector of one or more classify factors over which to split the comparisons. The same pairs set is tested, and adjusted, independently within each level (or combination of levels) of by, with no pooling across groups. Default NULL. See Details.
sig	The significance level for the confidence intervals, numeric between 0 and 1. Default is 0.05.
include_means	Logical; if TRUE (default) the predicted mean of each side of the comparison is included as level1.mean and level2.mean columns, immediately after estimate. Set FALSE for the differences only.
descending	Tri-state control of row ordering within each by-group. NULL (default) keeps the input order of pairs; FALSE sorts ascending by estimate; TRUE sorts descending by estimate.
x	A pairwise_comparisons object.
decimals	Number of decimal places to display. Default is 2. The p-value is shown to 3 significant figures (rather than rounded) so very small p-values do not collapse to zero.

Details

Relationship to the other comparison functions:

pairwise_comparisons(), multiple_comparisons() and reference_comparisons() share the same predicted means and standard errors of differences; they differ in which comparisons

they report. Testing all pairs here (`pairs = NULL`) with a given `adjust` yields the same adjusted p-values as `multiple_comparisons()` with that `adjust`, while `reference_comparisons()` is the special case of comparing every level against a single control. Tukey's HSD is exact only for the complete set of all pairwise comparisons, so `adjust = "tukey"` is not valid for `pairwise_comparisons()` (use `multiple_comparisons()`).

`pairs` syntax and sign convention:

The estimate for a pair is `mean(level1) - mean(level2)`, in the order written (`"A-B"` gives A - B). The `level1` and `level2` columns make the sign unambiguous. With `:` joining interaction cells and `-` separating the two sides of a pair, a level name may itself contain `-`: each `-` is tried as the separator and the split that yields two valid levels is used (so `"A-D-xyz"` resolves to A versus `D-xyz` when those are the real levels). Only when more than one split is valid is the pair genuinely ambiguous, and the list form is then required (the error shows the candidate splits). Reversed or duplicated pairs (`"A-B"` and `"B-A"`) are de-duplicated with a warning, since duplicates would inflate the adjustment family.

For an interaction `classify`, the `:`-joined components of each level label must be in the same order as `classify` (e.g. `"A:X"` for `classify = "Trt:Site"`, not `"X:A"`); the same applies to the level names used in `contrasts`. A label that names no existing cell is rejected with the list of available levels, but a mis-ordered label that happens to name another valid cell is used silently — so when the factors share level names, check the component order matches `classify`.

`by` semantics:

`by` must be a subset of the `classify` factors. Within each group, pair labels reference the remaining (non-by) factor levels. For example, `classify = "Trt:Site"`, `by = "Site"`, `pairs = "A-B"` compares Trt levels A and B within each Site. A group with fewer than two levels is skipped with a warning. In an unbalanced design a requested comparison may reference a level that is absent from some groups: such a comparison is skipped (with a warning) in the groups where it cannot be computed and reported in those where it can. A level that is absent from *every* group — or that was aliased — is instead an error, since that indicates a mistake rather than an incomplete design.

Standard error and degrees of freedom for `contrasts`:

The contrast variance is `c' V c`, where `V` is the variance-covariance matrix of the predicted means taken directly from the fitting engine (no reconstruction). The degrees of freedom depend on the engine:

- For models predicted via `emmeans` (`aov`, `lm`, `nlme::lme()`, `lme4::lmer()`, and `aov` with `Error()` strata) the estimate, standard error and degrees of freedom are obtained directly from `emmeans::contrast()` on the model's reference grid. The degrees of freedom are therefore the *exact* contrast df for that engine (Satterthwaite or Kenward-Roger for mixed models, containment for `aov`/`Error()`), including for contrasts spanning more than two levels.
- For `asreml` models `V` is the prediction error covariance from `predict(..., vcov = TRUE)`, and the degrees of freedom are the term's denominator df from `asreml::wald(denDF = "default")` (a single Kenward-Roger-style value shared by all contrasts within the term, as ASReml-R does not provide a per-contrast approximate df).

Transformations:

Comparisons are reported on the model scale. A difference of transformed means does not back-transform to a difference on the original scale (for log/logit it is a ratio), so back-transformation is not attempted; a warning is issued if the response appears to be transformed in the model formula.

Confidence intervals:

The per-comparison confidence intervals are *not* simultaneity-adjusted: as with the confidence-interval/letter note in `multiple_comparisons()`, an interval may exclude zero while the adjusted p-value is `>= sig`. When this happens a one-time `message()` notes it (suppressible with `suppressMessages()`).

Value

`autoplot.pairwise_comparisons()` returns a `ggplot2` object: a forest plot of the estimated differences with their confidence intervals and a dashed reference line at zero, faceted by the `by` variable(s) when present. Comparisons that are significant at the adjusted `sig` level are flagged with an asterisk (`*`) — prefixed to the y-axis label when unfaceted (keeping the labels right-justified against the axis), or beside the interval when faceted by `by`.

A `data.frame` of class `pairwise_comparisons` with one row per (group × comparison) and columns: any `by` column(s), `level1`, `level2`, `comparison`, `estimate`, (optionally `level1.mean` and `level2.mean`), `std.error`, `statistic`, `df`, `p.value` (adjusted), `conf.low` and `conf.high`. Stored at full precision; rounding for display is controlled by `print.pairwise_comparisons()`. For the `contrasts` form, the `level1`/`level2` and mean columns are omitted and `comparison` holds the contrast name. If any levels were aliased (not estimable) in the model they are dropped from the comparisons (with a warning) and recorded in an `aliased` attribute.

`print.pairwise_comparisons()` invisibly returns `x`.

Supported model types

The comparison functions (`multiple_comparisons()`, `pairwise_comparisons()` and `reference_comparisons()`) work with any model for which a `get_predictions()` method is defined. These are currently:

Model class	Fitted by	Notes
`aov`, `lm`	`stats::aov()`, `stats::lm()`	Fixed-effects linear models.
`aovlist`	`stats::aov()` with an `Error()` term	Multi-stratum aov; degrees of freedom are comparison-
`lme`	`nlme::lme()`	Linear mixed model.
`lmerMod`	`lme4::lmer()`, `lme4breeding::lmebreed()`	Linear mixed model. `lmebreed()` (relationship-based)
`lmerModLmerTest`	`lmerTest::lmer()`	As `lmerMod`, with Satterthwaite degrees of freedom.
`asreml`	ASReml-R `asreml()`	Linear mixed model (commercial; not on CRAN).
`afex_aov`	afex `aov_car()` / `aov_ez()` / `aov_4()`	Factorial / repeated-measures ANOVA; degrees of freed
`glmmTMB`	glmmTMB `glmmTMB()`	Generalized linear mixed model. Predictions are on the
`mmes`	sommer `mmes()`	Linear mixed model, via sommer's native `predict()`. S

ARTool (`art`) models are supported by `resplot()` but **not** by the comparison functions: the aligned rank transform makes mean-based comparisons inappropriate. Use `ARTool::art.con()` for contrasts on ART models instead.

sommer `mmer` models (the legacy interface) are supported by `resplot()` but **not** by the comparison functions: current sommer provides no `predict()` method for `mmer`. Refit with `sommer::mmes()` to use the comparison functions.

To add a new engine, write a `get_predictions.<class>()` method returning a list with elements `predictions`, `sed`, `df`, `ylab`, `aliased_names` and `classify` (plus `emmeans_grid` for engines backed by `emmeans::emmeans()`), and add a row to the table above.

ASReml-R terms in `classify`

For `asreml` models, `classify` names the factors to predict, as for ASReml-R `predict()`. A term wrapped in an ASReml-R function can be given either by its factors or as written in the model, so for a model with `at(Year):Treatment` both `classify = "Year:Treatment"` and `classify = "at(Year):Treatment"` give the same result. This applies to `at()` and to the variance-structure and relationship functions in the random model (e.g. `diag()`, `us()`, `fa()`, `vm()`), so `fa(Site, 2):Variety` is classified with `"Site:Variety"`. Covariates fitted with `pol()`, `spl()`, `lin()` and similar functions cannot be compared and are not accepted in `classify`.

Predictions of terms in the random model include the random effects (BLUPs). Their comparisons use the residual degrees of freedom, with a warning.

For an `at()` term, ASReml-R gives a separate Wald test, with its own denominator degrees of freedom, for each level of the conditioning factor. A comparison within a level uses that level's degrees of freedom; a comparison between levels uses the smaller of the two, as a conservative choice. Levels not included in `at()` use the residual degrees of freedom. A general contrast in `pairwise_comparisons()` uses the smallest degrees of freedom among the levels it involves.

ASReml-R prediction arguments

For `asreml` models, arguments given in `...` are passed to ASReml-R `predict()`. Those most useful for comparisons are:

- `present`: average only over the combinations of factor levels that occur in the data (see below).
- `average`: choose the factors to average over, optionally with weights (e.g. in proportion to replication rather than equally).
- `levels`: predict at chosen levels only, e.g. a subset of treatments, or at given values of a covariate rather than its mean.
- `ignore`, `use`, `except` and `only`: change which model terms enter the predictions, e.g. `use` to include a random term that is otherwise left out.
- `associate`: declare nested factors, e.g. treatments nested within treatment types.

`classify`, `sed` and `vcov` are set by the comparison functions and cannot be passed. `aliased = TRUE` (predictions of non-estimable functions) and `evaluate = FALSE` are not suitable. See ASReml-R `?predict.asreml` for full details of each argument.

When to use `present`. By default ASReml-R `predict()` averages over every combination of the levels of the fixed factors not in `classify`. If some combinations were never observed (treatments that differ between sites or years, a control outside a factorial set, or a factor fitted only within some levels of another, as with `at()`), the predictions that need them cannot be estimated: those levels are dropped as aliased, with a warning, or the function stops with "All predicted values are aliased". `present` restricts the averaging to the combinations in the data. Give it the factors involved, usually those in `classify` and those averaged over:

```
multiple_comparisons(model.asr, classify = "Year:Prior_crop",
                     present = c("Year", "Prior_crop", "Treatment"))
```

Each mean is then averaged over only the combinations observed for it, so two means can rest on different sets of levels of the other factors. Check that this is a fair basis for the comparison.

See Also

`multiple_comparisons()` for means-and-letters output, and `reference_comparisons()` for comparing every level against a single control. For guidance on choosing between them and on multiplicity adjustments, see `vignette("choosing-multiple-comparisons", "biometryassist")`.

Examples

```
# Forest plot of pairwise differences (significant comparisons marked with *)
dat.aov <- aov(Petal.Width ~ Species, data = iris)
pc <- pairwise_comparisons(dat.aov, classify = "Species")
autoplot(pc)
dat.aov <- aov(Petal.Width ~ Species, data = iris)

# All pairwise comparisons (Holm-adjusted by default)
pairwise_comparisons(dat.aov, classify = "Species")

# A selected subset
pairwise_comparisons(
    dat.aov,
    classify = "Species",
    pairs = c("setosa-versicolor", "setosa-virginica")
)

# A general (non-pairwise) contrast: setosa vs the average of the others
pairwise_comparisons(
    dat.aov,
    classify = "Species",
    contrasts = list(
        "setosa vs rest" = c(setosa = 1, versicolor = -0.5, virginica = -0.5)
    )
)

# `by`: the same comparisons, adjusted independently within each group. Here a
# 2 x 3 factorial - compare tension levels within each wool type. The `pairs`
# labels reference the remaining (tension) factor.
m_wb <- aov(breaks ~ wool * tension, data = warpbreaks)
pairwise_comparisons(
    m_wb,
    classify = "wool:tension",
    by = "wool",
    pairs = c("L-M", "L-H")
)
```

`autoplot.reference_comparisons` *Compare predicted means against a single reference (control) level*

Description

Compare every level of a treatment factor against one chosen **reference** (control) level, returning a tidy, means-centric table: the predicted mean of each level, the reference mean, their difference, and a multiplicity-adjusted p-value for the difference. This is the honest representation of the common "how does each treatment compare to the control?" question (a Dunnett-style analysis), where significance attaches cleanly to each treatment because every comparison shares the same reference.

Usage

```
## S3 method for class 'reference_comparisons'
autoplot(object, ..., axis_rotation = 0, label_rotation = 0)

reference_comparisons(
  model.obj,
  classify,
  reference,
  adjust = "dunnett",
  by = NULL,
  sig = 0.05,
  include_means = TRUE,
  descending = NULL,
  ...
)

## S3 method for class 'reference_comparisons'
print(x, decimals = 2, ...)
```

Arguments

object	A reference_comparisons object.
...	Other arguments passed to the model-specific prediction methods (e.g. ASReml-R predict() arguments such as present; see *ASReml-R prediction arguments* below).
axis_rotation	Rotation (degrees) of the x-axis (mean) labels.
label_rotation	Rotation (degrees) of the y-axis (level) labels.
model.obj	An asreml, aov, lm, lme (nlme::lme()) or lmerMod (lme4::lmer()) model object.
classify	Name of the predictor variable(s) to compare, as a string. Interactions are specified with : (e.g. "Trt:Site"). For asreml models, see *ASReml-R terms in* classify below.
reference	The reference (control) level to compare every other level against, as a single character string. When by is used it is a level of the remaining (non-by) factor; for an interaction classify with no by it is the full :-joined cell label, whose components must be in the same order as classify (e.g. "A:X" for classify = "Trt:Site"). Each comparison is mean(level) - mean(reference), so a positive estimate is "above the control". See Details for guidance on interaction classify.

adjust	The method used to adjust p-values for multiplicity over the set of comparisons against the reference. Default "dunnett" performs the exact simultaneous two-sided Dunnett test (via the multivariate-t distribution). Any stats::p.adjust.methods value is also accepted (e.g. "holm", "bonferroni", "BH"); "tukey" is **not** valid here (use multiple_comparisons()). See Details.
by	A character vector of one or more classify factors over which to split the comparisons. The same reference set is tested, and adjusted, independently within each level (or combination of levels) of by. Default NULL. See Details.
sig	The significance level for the confidence intervals, numeric between 0 and 1. Default is 0.05.
include_means	Logical. The predicted means are central to this display, so they are always included (level1.mean, level2.mean); setting FALSE is ignored with a warning.
descending	Tri-state control of row ordering within each by-group. NULL (default) keeps prediction order; FALSE sorts ascending by estimate; TRUE sorts descending by estimate. This orders by the comparison *estimate* (each level minus the reference), unlike multiple_comparisons() which orders by the predicted *mean*.
x	A reference_comparisons object.
decimals	Number of decimal places to display. Default is 2. The p-value is shown to 3 significant figures (rather than rounded) so very small p-values do not collapse to zero.

Details

Unlike multiple_comparisons() (all pairs, means + letters) and pairwise_comparisons() (selected differences), reference_comparisons() compares each level to a single control and presents the result around the means. It works for every model supported by multiple_comparisons().

Why Dunnett is the default:

Comparing several treatments to one control has an *exact* simultaneous procedure — Dunnett's test — which uses the joint multivariate-t distribution of the comparisons (accounting for the correlation induced by the shared control). It is more powerful than applying a generic adjustment, so it is the default. The exact test requires a single (common) degrees-of-freedom; for models that report comparison-specific df, reference_comparisons() falls back to "holm" with a warning. Any stats::p.adjust.methods method may be requested explicitly (a message notes that Dunnett is the exact option).

The multivariate-t routine (mvtnorm::pmvt()) requires an integer degrees-of-freedom, so a *fractional* denominator df (such as an ASReml-R Kenward-Roger denDF) is rounded to the nearest integer for the Dunnett calculation when two or more comparisons are made. The reported df column is the exact (unrounded) value, and a single comparison uses the exact df. In practice this rounding only affects asreml models with a fractional denDF (aov/lm/lme have integer df, and mixed models with comparison-specific df fall back to Holm).

The Dunnett correlation structure is taken from the variance-covariance matrix of the predicted means that the prediction machinery returns, so the exact test is available for every supported model engine. With adjust = "dunnett" the confidence intervals are the *simultaneous* Dunnett intervals and therefore agree with the adjusted test (an interval excludes zero exactly when the comparison is significant). With a stats::p.adjust() method the intervals are per-comparison and may disagree with the adjusted p-value.

by **semantics:**

by must be a subset of the classify factors. Within each group, the reference and the compared levels reference the remaining (non-by) factor. For example classify = "Trt:Site", by = "Site", reference = "Control" compares every Trt level against Control *within each Site*, adjusted within each Site. A group missing the reference, or with fewer than two levels, is skipped with a warning.

To compare a control level against the others within every combination of the *remaining* factors, put all the other factors in by. The within-group factor is then the single control factor, so reference is just that level (e.g. "0"). For example, with classify = "time:variety:dose" and a zero dose labelled "0", by = c("time", "variety"), reference = "0" compares each dose against the zero dose within every time-by-variety cell. Specified this way reference is a single level, so it is unaffected by the order of the factors in classify or by (those orderings only change the column order and group labels of the output, not the comparisons).

Without by, by contrast, reference is the full :-joined cell label and its components must follow the classify order. A mis-ordered label that happens to name another valid cell is used silently (no error), so it is worth double-checking the order matches; the order-proof by form above avoids this entirely.

Transformations:

Comparisons are reported on the model scale (a difference of transformed means does not back-transform to a difference on the original scale); a warning is issued if the response appears to be transformed in the model formula.

Value

autoplot.reference_comparisons() returns a ggplot2 object: a means plot with one point per level at its predicted mean, a dashed reference line at the reference mean (marked with a diamond), and an interval around each mean showing the (adjusted) confidence interval for the difference from the reference — so the interval clears the reference line exactly when the comparison is significant (with adjust = "dunnett"). Faceted by the by variable(s) when present. Significant comparisons are flagged with an asterisk (*), prefixed to the y-axis label when unfaceted or beside the interval when faceted.

A data.frame of class reference_comparisons with one row per (group × non-reference level) and columns: any by column(s), level1, level2 (the reference), comparison, estimate, level1.mean, level2.mean (the reference mean), std.error, statistic, df, p.value (adjusted), conf.low and conf.high. The reference level is not given its own row (its mean appears as level2.mean on every row, and in the reference attribute). Stored at full precision; rounding for display is controlled by print.reference_comparisons(). If any levels were aliased (not estimable) in the model they are dropped (with a warning) and recorded in an aliased attribute; an aliased reference is an error.

print.reference_comparisons() invisibly returns x.

Supported model types

The comparison functions (multiple_comparisons(), pairwise_comparisons() and reference_comparisons()) work with any model for which a get_predictions() method is defined. These are currently:

Model class	Fitted by	Notes
aov, lm	stats::aov(), stats::lm()	Fixed-effects linear models.
aovlist	stats::aov() with an Error() term	Multi-stratum aov; degrees of freedom are comparison-
lme	nlme::lme()	Linear mixed model.
lmerMod	lme4::lmer(), lme4breeding::lmebreed()	Linear mixed model. lmebreed() (relationship-based)
lmerModLmerTest	lmerTest::lmer()	As lmerMod, with Satterthwaite degrees of freedom.
asreml	ASReml-R asreml()	Linear mixed model (commercial; not on CRAN).
afex_aov	afex aov_car() / aov_ez() / aov_4()	Factorial / repeated-measures ANOVA; degrees of freed
glmmTMB	glmmTMB glmmTMB()	Generalized linear mixed model. Predictions are on the
mmes	sommer mmes()	Linear mixed model, via sommer's native predict(). S

ARTool (art) models are supported by resplot() but **not** by the comparison functions: the aligned rank transform makes mean-based comparisons inappropriate. Use ARTool::art.con() for contrasts on ART models instead.

sommer mmer models (the legacy interface) are supported by resplot() but **not** by the comparison functions: current sommer provides no predict() method for mmer. Refit with sommer::mmes() to use the comparison functions.

To add a new engine, write a get_predictions.<class>() method returning a list with elements predictions, sed, df, ylab, aliased_names and classify (plus emmeans_grid for engines backed by emmeans::emmeans()), and add a row to the table above.

ASReml-R terms in classify

For asreml models, classify names the factors to predict, as for ASReml-R predict(). A term wrapped in an ASReml-R function can be given either by its factors or as written in the model, so for a model with at(Year):Treatment both classify = "Year:Treatment" and classify = "at(Year):Treatment" give the same result. This applies to at() and to the variance-structure and relationship functions in the random model (e.g. diag(), us(), fa(), vm()), so fa(Site, 2):Variety is classified with "Site:Variety". Covariates fitted with pol(), spl(), lin() and similar functions cannot be compared and are not accepted in classify.

Predictions of terms in the random model include the random effects (BLUPs). Their comparisons use the residual degrees of freedom, with a warning.

For an at() term, ASReml-R gives a separate Wald test, with its own denominator degrees of freedom, for each level of the conditioning factor. A comparison within a level uses that level's degrees of freedom; a comparison between levels uses the smaller of the two, as a conservative choice. Levels not included in at() use the residual degrees of freedom. A general contrast in pairwise_comparisons() uses the smallest degrees of freedom among the levels it involves.

ASReml-R prediction arguments

For asreml models, arguments given in ... are passed to ASReml-R predict(). Those most useful for comparisons are:

- present: average only over the combinations of factor levels that occur in the data (see below).
- average: choose the factors to average over, optionally with weights (e.g. in proportion to replication rather than equally).

- `levels`: predict at chosen levels only, e.g. a subset of treatments, or at given values of a covariate rather than its mean.
- `ignore`, `use`, `except` and `only`: change which model terms enter the predictions, e.g. `use` to include a random term that is otherwise left out.
- `associate`: declare nested factors, e.g. treatments nested within treatment types.

`classify`, `sed` and `vcov` are set by the comparison functions and cannot be passed. `aliased = TRUE` (predictions of non-estimable functions) and `evaluate = FALSE` are not suitable. See ASReml-R `?predict.asreml` for full details of each argument.

When to use `present`. By default ASReml-R `predict()` averages over every combination of the levels of the fixed factors not in `classify`. If some combinations were never observed (treatments that differ between sites or years, a control outside a factorial set, or a factor fitted only within some levels of another, as with `at()`), the predictions that need them cannot be estimated: those levels are dropped as aliased, with a warning, or the function stops with "All predicted values are aliased". `present` restricts the averaging to the combinations in the data. Give it the factors involved, usually those in `classify` and those averaged over:

```
multiple_comparisons(model.asr, classify = "Year:Prior_crop",
                     present = c("Year", "Prior_crop", "Treatment"))
```

Each mean is then averaged over only the combinations observed for it, so two means can rest on different sets of levels of the other factors. Check that this is a fair basis for the comparison.

See Also

`multiple_comparisons()` for all-pairs means and letters, `pairwise_comparisons()` for selected differences. For guidance on choosing between them and on multiplicity adjustments, see `vignette("choosing-multiple-c`
`"biometryassist")`.

Examples

```
# Means plot of each level vs the reference (significant ones marked with *)
dat.aov <- aov(Petal.Width ~ Species, data = iris)
rc <- reference_comparisons(dat.aov, classify = "Species", reference = "setosa")
autoplot(rc)
dat.aov <- aov(weight ~ feed, data = chickwts)

# Compare every feed against the "casein" control (exact Dunnett)
reference_comparisons(dat.aov, classify = "feed", reference = "casein")

# A different adjustment can be requested (a message notes that Dunnett is the
# exact option for this family):
reference_comparisons(
    dat.aov,
    classify = "feed",
    reference = "casein",
    adjust = "holm"
)

# `by`: compare each level against the reference *within* each group, adjusted
```

```
# independently. Here a 2 x 3 factorial - each tension level vs the "L"
# reference within each wool type; autoplot() facets by wool.
m_wb <- aov(breaks ~ wool * tension, data = warpbreaks)
rc_by <- reference_comparisons(
    m_wb,
    classify = "wool:tension",
    reference = "L",
    by = "wool"
)
rc_by
autoplot(rc_by)
```

`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_numbers = FALSE,
  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), "double column" (a buffer row each side of a treatment column), "blocks" (buffers at internal boundaries between adjacent blocks), or "double block"/"entire block"/"full block" (buffers fully surrounding each block).
plot_numbers	One of FALSE (default), TRUE/"sequential", or "serpentine". Controls whether a plot_number column is added to the design. "sequential" (or TRUE) numbers plots row-by-row from top-left to bottom-right; "serpentine" reverses direction on alternate rows, following typical field navigation paths. Plot numbers are assigned after any buffers are added, so buffer plots are numbered alongside treatment plots.
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,
  filename = "experimental_design.xlsx",
  value_column = "treatments",
  row = NULL,
  column = NULL,
  palette = "default"
)
```

Arguments

design	A dataframe or design object containing experimental design.
filename	Character string for Excel filename (default: "experimental_design.xlsx")
value_column	Character string specifying which column to use for layout values (default: "treatments")
row	Column containing the row coordinate. Defaults to row.
column	Column containing the column coordinate. Defaults to col.
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

```
my_design <- design("crd", treatments = c("A", "B", "C"), reps = 2,
                    nrows = 2, ncols = 3, quiet = TRUE, plot = FALSE)
file <- tempfile(fileext = ".xlsx")
```

```
# Export with default colours
export_design_to_excel(my_design, file)

# Export without colours
export_design_to_excel(my_design, file, palette = NULL)

# Export with custom palette
export_design_to_excel(my_design, file, palette = "viridis")
```

---

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,
  adjust = "tukey",
  by = NULL,
  plot = FALSE,
  label_height = 0.1,
  rotation = 0,
  save = FALSE,
  savename = "predicted_values",
  ...
)
```

Arguments

model.obj	An asreml, aov, lm, lme (nlme::lme()) or lmerMod (lme4::lmer()) model object.
classify	Name of predictor variable as string. Interactions are specified with : (e.g. "Trt:Site"). For asreml models, see *ASReml-R terms in* classify below.
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	Deprecated. Rounding is now controlled via the decimals argument of print.mct().
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.
adjust	The method used to adjust p-values for multiple comparisons. Either "tukey" (default, Tukey's HSD) or any method accepted by stats::p.adjust() ("bonferroni", "holm", "hochberg", "hommel", "BH" (or "fdr"), "BY", or "none"). See Details.
by	A character vector of column name(s) in the predictions over which to split comparisons. Comparisons are run independently within each level (or combination of levels) of the by variable(s); no p-values are pooled or adjusted across groups. Default NULL. See Details.
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.
...	Other arguments passed to the model-specific prediction methods (e.g. ASReml-R predict() arguments such as present; see *ASReml-R prediction arguments* below).

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.

P-value adjustment (`adjust`):

By default (`adjust = "tukey"`) the function uses Tukey's HSD, which is exact for the complete set of all pairwise comparisons. Alternatively, `adjust` may be any method accepted by `stats::p.adjust()`. In that case a matrix of *raw* two-sided t-test p-values is computed first and the chosen adjustment is applied to the lower-triangle vector of those raw p-values, avoiding the double-adjustment that would occur if Tukey-adjusted p-values were passed to `p.adjust()`. Bonferroni and Holm control the family-wise error rate and are valid under arbitrary dependence; BH/BY (FDR) are valid under positive dependence, which pairwise comparisons satisfy. The returned `$pairwise_pvalues` always holds the adjusted p-values for the chosen method (the Tukey p-values when `adjust = "tukey"`). When `adjust` is not `"tukey"`, `$hsd` is `NULL`, as an HSD value is only meaningful for Tukey's test.

Grouped comparisons (`by`):

When `by` is supplied, the predictions are split into subgroups defined by the level(s) of the `by` variable(s) and the comparison procedure is run independently within each subgroup. Each subgroup is treated as a separate family of comparisons: there is no pooling and no cross-group p-value adjustment, and letter groupings restart within each subgroup. When `by` is used, `$pairwise_pvalues` (and `$hsd` where applicable) are returned as named lists with one element per subgroup, and `autoplot()` facets the plot by the `by` variable(s) by default. `by` must leave at least one `classify` factor to compare within each subgroup, so a single-factor `classify` cannot be split with `by`.

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 adjusted p-values for all pairwise comparisons (Tukey's HSD by default, otherwise adjusted by the `adjust` method). When `by` is supplied, a named list of such matrices, one per subgroup
`hsd`	The Honest Significant Difference value(s) used in the comparisons when `adjust = "tukey"`. Either a single numeric value (if constant across comparisons) or a matrix (if it varies by comparison). `NULL` when `adjust` is not `"tukey"`
`sig_level`	The significance level used (default 0.05)
`comparison_method`	The p-value adjustment method used (the value of `adjust`)
`aliased`	Character vector of aliased treatment levels (only present if some predictions are aliased)

Supported model types

The comparison functions (`multiple_comparisons()`, `pairwise_comparisons()` and `reference_comparisons()`) work with any model for which a `get_predictions()` method is defined. These are currently:

Model class	Fitted by	Notes
`aov`, `lm`	`stats::aov()`, `stats::lm()`	Fixed-effects linear models.

aovlist	stats::aov() with an Error() term	Multi-stratum aov; degrees of freedom are comparison-
lme	nlme::lme()	Linear mixed model.
lmerMod	lme4::lmer(), lme4breeding::lmebreed()	Linear mixed model. lmebreed() (relationship-based)
lmerModLmerTest	lmerTest::lmer()	As lmerMod, with Satterthwaite degrees of freedom.
asreml	ASReml-R asreml()	Linear mixed model (commercial; not on CRAN).
afex_aov	afex aov_car() / aov_ez() / aov_4()	Factorial / repeated-measures ANOVA; degrees of freed
glmmTMB	glmmTMB glmmTMB()	Generalized linear mixed model. Predictions are on the
mmes	sommer mmes()	Linear mixed model, via sommer's native predict(). S

ARTool (art) models are supported by resplot() but **not** by the comparison functions: the aligned rank transform makes mean-based comparisons inappropriate. Use ARTool::art.con() for contrasts on ART models instead.

sommer mmer models (the legacy interface) are supported by resplot() but **not** by the comparison functions: current sommer provides no predict() method for mmer. Refit with sommer::mmes() to use the comparison functions.

To add a new engine, write a get_predictions.<class>() method returning a list with elements predictions, sed, df, ylab, aliased_names and classify (plus emmeans_grid for engines backed by emmeans::emmeans()), and add a row to the table above.

ASReml-R terms in classify

For asreml models, classify names the factors to predict, as for ASReml-R predict(). A term wrapped in an ASReml-R function can be given either by its factors or as written in the model, so for a model with at(Year):Treatment both classify = "Year:Treatment" and classify = "at(Year):Treatment" give the same result. This applies to at() and to the variance-structure and relationship functions in the random model (e.g. diag(), us(), fa(), vm()), so fa(Site, 2):Variety is classified with "Site:Variety". Covariates fitted with pol(), spl(), lin() and similar functions cannot be compared and are not accepted in classify.

Predictions of terms in the random model include the random effects (BLUPs). Their comparisons use the residual degrees of freedom, with a warning.

For an at() term, ASReml-R gives a separate Wald test, with its own denominator degrees of freedom, for each level of the conditioning factor. A comparison within a level uses that level's degrees of freedom; a comparison between levels uses the smaller of the two, as a conservative choice. Levels not included in at() use the residual degrees of freedom. A general contrast in pairwise_comparisons() uses the smallest degrees of freedom among the levels it involves.

ASReml-R prediction arguments

For asreml models, arguments given in ... are passed to ASReml-R predict(). Those most useful for comparisons are:

- present: average only over the combinations of factor levels that occur in the data (see below).
- average: choose the factors to average over, optionally with weights (e.g. in proportion to replication rather than equally).

- `levels`: predict at chosen levels only, e.g. a subset of treatments, or at given values of a covariate rather than its mean.
- `ignore`, `use`, `except` and `only`: change which model terms enter the predictions, e.g. `use` to include a random term that is otherwise left out.
- `associate`: declare nested factors, e.g. treatments nested within treatment types.

`classify`, `sed` and `vcov` are set by the comparison functions and cannot be passed. `aliased = TRUE` (predictions of non-estimable functions) and `evaluate = FALSE` are not suitable. See ASReml-R `?predict.asreml` for full details of each argument.

When to use `present`. By default ASReml-R `predict()` averages over every combination of the levels of the fixed factors not in `classify`. If some combinations were never observed (treatments that differ between sites or years, a control outside a factorial set, or a factor fitted only within some levels of another, as with `at()`), the predictions that need them cannot be estimated: those levels are dropped as aliased, with a warning, or the function stops with "All predicted values are aliased". `present` restricts the averaging to the combinations in the data. Give it the factors involved, usually those in `classify` and those averaged over:

```
multiple_comparisons(model.asr, classify = "Year:Prior_crop",
                     present = c("Year", "Prior_crop", "Treatment"))
```

Each mean is then averaged over only the combinations observed for it, so two means can rest on different sets of levels of the other factors. Check that this is a fair basis for the comparison.

References

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

See Also

`pairwise_comparisons()` for testing a chosen subset of pairwise differences as a tidy table, or `reference_comparisons()` for testing treatments against a chosen reference or control. For guidance on choosing between the two and on multiplicity adjustments, see `vignette("choosing-multiple-comparisons", "biometryassist")`.

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)

# Use a different p-value adjustment instead of Tukey's HSD
multiple_comparisons(model, classify = "Species", adjust = "fdr")

# `by`: run the comparisons independently within each level of another factor.
# Here a 2 x 3 factorial - compare tension levels within each wool type.
m_wb <- aov(breaks ~ wool * tension, data = warpbreaks)
mc_by <- multiple_comparisons(m_wb, classify = "wool:tension", by = "wool")
mc_by
autoplot(mc_by) # faceted by wool

# 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-R 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)

print(pred.out, decimals = 5)

# 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, decimals = 2, ...)
```

Arguments

x	An mct object to print to the console.
decimals	Number of decimal places to display. Default is 2.
...	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)
print(output, decimals = 4)
```

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	A fitted model object of a supported class (aov, lm, lme (nlme::lme()), lmerMod (lme4::lmer()), asreml, mmer/mmes (sommer) or art (ARTool)). See the *Supported model types* section.
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.

Supported model types

resplot() produces residual diagnostics for any model with an extract_model_info() method. These are currently:

Model class	Fitted by	Notes
aov, lm	stats::aov(), stats::lm()	Fixed-effects linear models.
aovlist	stats::aov() with an Error() term	Multi-stratum aov; each error stratum (except the interc
lme	nlme::lme()	Linear mixed model.
lmerMod	lme4::lmer(), lme4breeding::lmebreed()	Linear mixed model. lmebreed() (relationship-based)
lmerModLmerTest	lmerTest::lmer()	As lmerMod.
asreml	ASReml-R asreml()	Linear mixed model (commercial; not on CRAN). Resi

mmer, mmes	sommer mmer() / mmes()	Linear mixed model.
art	ARTool::art()	Aligned rank transform model.
afex_aov	afex aov_car() / aov_ez() / aov_4()	Factorial / repeated-measures ANOVA; a single diagnos
glmmTMB	glmmTMB glmmTMB()	**Gaussian family only.** Non-Gaussian families error wi

This set differs slightly from the comparison functions (see get_predictions()): resplot() additionally supports ARTool (art) models and sommer's legacy mmer interface. Neither is available for the comparison functions — ART uses aligned ranks (use ARTool::art.con()), and current sommer provides no predict() for mmer (refit with sommer::mmes()).

To add a new engine, write an extract_model_info.<class>() method returning a list with elements facet, facet_name, resids, fits, k and model_call, and add a row to the table above.

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("aov_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

add_buffers, 2
autoplot, 4
autoplot(), *20*
autoplot.design, 4
autoplot.design(), *4*
autoplot.mct, 6
autoplot.mct(), *4*, *29*
autoplot.pairwise_comparisons, 8
autoplot.pairwise_comparisons(), *4*
autoplot.reference_comparisons, 13
autoplot.reference_comparisons(), *4*

design, 19
design(), *4, 5*

emmeans::contrast(), *10*
emmeans::emmeans(), *11*, *17*, *32*
export_design_to_excel, 22

get_predictions(), *38*
ggplot2::autoplot(), *4*
ggplot2::element_text(), *7*
ggplot2::facet_wrap(), *24*
ggplot2::geom_raster(), *24*
ggplot2::geom_text(), *5*
ggplot2::ggsave(), *21*
grDevices::hcl.pals(), *24*

heat_map, 24

install_asreml, 25

list_templates, 27
list_templates(), *40*
lme4::lmer(), *9–11*, *14*, *17*, *29*, *32*, *37*
lmerTest::lmer(), *11*, *17*, *32*, *37*
logl_test, 27

multiple_comparisons, 28
multiple_comparisons(), *4*, *6–11*, *13*, *15, 16*, *18*, *31*, *36*
mvtnorm::pmvt(), *15*

nlme::lme(), *9–11*, *14*, *17*, *29*, *32*, *37*

pairwise_comparisons *(autoplot.pairwise_comparisons)*, 8
pairwise_comparisons(), *11, 12*, *15–18*, *31–33*
print.mct, 36
print.mct(), *29*
print.pairwise_comparisons *(autoplot.pairwise_comparisons)*, 8
print.pairwise_comparisons(), *11*
print.reference_comparisons *(autoplot.reference_comparisons)*, 13
print.reference_comparisons(), *16*

reference_comparisons *(autoplot.reference_comparisons)*, 13
reference_comparisons(), *9–11*, *13*, *16*, *31*, *33*
resplot, 36
resplot(), *11*, *17*, *32*, *37*

scales::brewer_pal(), *5*, *41*
stats::aov(), *11*, *17*, *31, 32*, *37*
stats::lm(), *11*, *17*, *31*, *37*
stats::p.adjust(), *9*, *15*, *29, 30*
stats::p.adjust.methods, *9*, *15*
summary_graph, 38
suppressMessages(), *11*

update_asreml *(install_asreml)*, 25
use_template, 39
use_template(), *27*

variogram, 40