Package: CBADASReml (via r-universe)

May 28, 2026

Title Provides helper functions for ASReml

Version 0.2.1

Description This package provides helper functions for ASReml, for use with small-plot and on-farm experimentation trials.

License GPL (>= 3)

URL https://aagi-aus.github.io/CBADASReml/

Depends R (>= 4.1.0)

Imports asremlPlus, car, cowplot, cli, dplyr, emmeans, ggplot2, gstat, rlang, sf

Suggests asreml, agricolae, agridat, glmmTMB, knitr, rmarkdown, roxyglobals, testthat (>= 3.0.0), vdiffr

VignetteBuilder knitr

Config/roxygen2/version 8.0.0

Config/roxyglobals/filename globals.R

Config/roxyglobals/unique FALSE

Config/testthat/edition 3

Config/testthat/parallel true

Encoding UTF-8

Language en-AU

Roxygen list(markdown = TRUE, roclets = c(``collate", ``namespace", ``rd", ``roxyglobals::global_roclet"))

X-schema.org-applicationCategory Tools

X-schema.org-isPartOf https://grdc.com.au/research/partnerships-and-initiatives/strategic-partnerships/aagi

X-schema.org-keywords Agricultural Research, Experimental Analysis

Config/pak/sysreqs libabsl-dev cmake libfftw3-dev libfontconfig1-dev libfreetype6-dev libfribidi-dev libgdal-dev gdal-bin libgeos-dev git make libharfbuzz-dev libgit2-dev libicu-dev libjpeg-dev libpng-dev libtiff-dev libuv1-dev libwebp-dev libxml2-dev libssl-dev libproj-dev libsqlite3-dev libudunits2-dev libx11-dev zlib1g-dev

Repository https://aagi-aus.r-universe.dev

Date/Publication 2026-05-28 02:32:54 UTC

RemoteUrl https://github.com/AAGI-AUS/CBADASReml

RemoteRef HEAD

RemoteSha 4479effbcfdcb45862f4731dcabf16ee4d0ee062

Contents

`anova_table` *An ANOVA table function*

Description

This function allows you to observe the ANOVA table for multiple **ASReml** or **glmmTMB** models.

Usage

```
anova_table(..., n_digits = 3)
```

Arguments

`...`	The models to use in the ANOVA table comparison Their values may be: • `asreml` • `glmmTMB`
`n_digits`	`numeric` The number of digits to round results to.

Value

`data.frame` A dataframe containing all ANOVA tables. Can be used with xtable to produce a LaTeX table.

Examples

```
library(CBADASReml)
library(asreml)
test_data <- oats
test_data["yield2"] <- oats["yield"] * runif(nrow(oats["yield"]))
mod1 <- asreml(
    fixed = yield ~ Variety + Nitrogen + Variety:Nitrogen,
    random = ~ idv(Blocks) + idv(Blocks):idv(Wplots),
    residual = ~ idv(units),
    data = test_data
)
mod2 <- asreml(
    fixed = yield2 ~ Variety + Nitrogen + Variety:Nitrogen,
    random = ~ idv(Blocks) + idv(Blocks):idv(Wplots),
    residual = ~ idv(units),
    data = test_data
)
anova_table(mod1, mod2)

## With glmmTMB models
library(glmmTMB)
Salamanders$count2 <- runif(nrow(Salamanders), 0, 10)
mod3 <- glmmTMB( # Zero inflated model
    count ~ spp * mined + (1 | site),
    zi = ~ spp * mined,
    data = Salamanders,
    family = nbinom2
)
mod4 <- glmmTMB( # Hurdle model
    count2 ~ spp * mined + (1 | site),
    zi = ~ spp * mined,
    data = Salamanders,
    family = truncated_nbinom2
)
anova_table(mod3, mod4)
```

---

design_efficiency *Calculate the design efficiency/optimality of a small-plot trial design*

Description

Calculates model-based efficiency for a design using the treatment information matrix induced by an AR1 x AR1 spatial dependence structure. All this function does is evaluate how much information the design contains about treatment contrasts, after adjusting for nuisance effects (e.g. blocks , etc.)

Usage

```
design_efficiency(
  design,
  treatment_cols = "treatment",
  row_col = "row",
  column_col = "col",
  block_col = NULL,
  rho_row = 0.1,
  rho_col = 0.1,
  alpha = 0.05,
  tolerance = 0.0000000001
)
```

Arguments

design	data.frame. The design data frame, one row per experimental unit.
treatment_cols	character vector. Column name of the treatment in design. If multiple columns are provided, each unique combination is treated as a distinct treatment combination.
row_col	character vector. Column name of the experiment's rows in design.
column_col	character vector. Column name of the experiment's columns in design.
block_col	character vector or NULL. Column name of the experiment's blocking factor in design. Use NULL for unblocked designs.
rho_row	numeric. AR1 correlation parameter in the row direction.
rho_col	numeric. AR1 correlation parameter in the column direction.
alpha	significance level numeric.
tolerance	numeric. Tolerance for numerical stability.

Details

The function computes the treatment information matrix

$$I_T = X_1^\top L_R X_1,$$

where X_1 is the treatment indicator matrix and L_R is the covariance-adjusted projection matrix that removes nuisance effects:

$$L = \Sigma^{-1} - \Sigma^{-1} X_2 \left(X_2^\intercal \Sigma^{-1} X_2 \right)^{-1} \Sigma^{-1}$$

Pairwise treatment contrast variances are then calculated from I_T^+, the Moore-Penrose pseudoinverse of the treatment information matrix.

Value

A list containing:

fisher_info The treatment information matrix, I_T.

eigenvalues Positive eigenvalues of the treatment information matrix.

rank Numerical rank of the treatment information matrix.

estimable_design Whether all treatment contrasts are estimable. For v treatment levels, this requires rank at least $v - 1$.

a_optimality A-optimality score, calculated as the sum of inverse positive eigenvalues. Smaller values indicate better average treatment contrast precision.

d_optimality D-optimality score, calculated as the negative sum of log positive eigenvalues. Smaller values indicate greater overall information volume.

pairwise_efficiency Data frame containing pairwise treatment comparisons, estimability indicators, contrast variances, and iid-equivalent effective replicates.

assumptions List of spatial correlation and design assumptions used in the calculation.

A list with a lot of info in it.

Examples

```
## Not run:
# RCBD
df <- data.frame(
    row = rep(1:6, each = 4),
    col = rep(1:4, times = 6),
    treatment = rep(LETTERS[1:8], 3),
    block = rep(1:3, each = 8)
)

# Optimise while respecting blocks
result <- speed::speed(df,
    "treatment",
    swap_within = "block",
    iterations = 5000,
    seed = 42
)

## test on latin square
latinsquare <- data.frame(
    row = rep(1:4, each = 4),
    col = rep(1:4, times = 4),
    treatment = c(
        "A", "B", "C", "D",
        "B", "C", "D", "A",
        "C", "D", "A", "B",
        "D", "A", "B", "C"
    )
)

res_eff <- design_efficiency(
    design = latinsquare,
    treatment_cols = "treatment",
    row_col = "row",
    column_col = "col",
```

```
    block_col = NULL,
    rho_row = 0.3,
    rho_col = 0.3
)

## End(Not run)
```

`design_power` *Calculate the statistical power for an experimental design*

Description

Calculate pairwise treatment-comparison power from the Fisher information of the provided experimental design using the treatment information matrix induced by an AR1 x AR1 spatial dependence structure. This function evaluates how much "power" the design has to detect a specified treatment difference after adjusting for nuisance effects (e.g. blocks, etc.)

Usage

```
design_power(
  design,
  treatment_cols = "treatment",
  row_col = "row",
  column_col = "col",
  block_col = NULL,
  rho_row = 0.1,
  rho_col = 0.1,
  sigma2 = 1,
  delta = 1,
  alpha = 0.05,
  tolerance = 0.0000000001
)
```

Arguments

`design`	`data.frame`. The design data frame, one row per experimental unit.
`treatment_cols`	`character` vector. Column name of the treatment in `design`. If multiple columns are provided, each unique combination is treated as a distinct treatment combination.
`row_col`	`character` vector. Column name of the experiment's rows in `design`.
`column_col`	`character` vector. Column name of the experiment's columns in `design`.
`block_col`	`character` vector or `NULL`. Column name of the experiment's blocking factor in `design`. Use `NULL` for unblocked designs.
`rho_row`	`numeric`. AR1 correlation parameter in the row direction.
`rho_col`	`numeric`. AR1 correlation parameter in the column direction.

`sigma2`	`numeric`. Assumed residual variance.
`delta`	`numeric`. Treatment difference that is "worth detecting". The minimum difference between treatments that we care about.
`alpha`	`numeric`. Significance/p-value threshold for the two-sided z-tests.
`tolerance`	`numeric`. Tolerance for numerical stability.

Details

The calculation is based on the treatment information matrix

$$I_T = X_1^\intercal L_R X_1,$$

where X_1 is the treatment indicator matrix and L_R is the covariance-adjusted projection matrix that removes nuisance effects:

$$L = \Sigma^{-1} - \Sigma^{-1} X_2 \left(X_2^\intercal \Sigma^{-1} X_2 \right)^{-1} \Sigma^{-1}$$

For a treatment contrast $c^\intercal \tau$, the contrast standard error is computed from

$$\mathrm{SE}(c^\intercal \hat{\tau}) = \sqrt{\sigma^2 c^\intercal I_T^+ c},$$

where I_T^+ is the Moore-Penrose pseudoinverse of the treatment information matrix. The effect size `delta` is then converted into a noncentrality parameter,

$$\lambda = \frac{\delta}{\mathrm{SE}(c^\intercal \hat{\tau})},$$

and power is calculated using a two-sided known-covariance Z-test approximation.

The spatial covariance parameters `rho_row`, `rho_col`, and `sigma2` are treated as assumed planning values, not estimated from the data. The returned power is therefore conditional on the supplied covariance structure, the chosen effect size `delta`, and the significance level `alpha`.

Thus, you as the statistician must select four parameters based on previous trials, domain knowledge, or *conservative* assumptions:

- ρ_{row} AR1 row dependence parameter. How correlated are observations that are one row apart? This is hard to select, I would recommend testing multiple different values such as 0.0 for no dependence, 0.3 for moderate, and 0.5 for strong dependence, and reporting each.
- ρ_{col} AR1 column dependence parameter. How correlated are observations that are one column apart? This is hard to select, I would recommend testing multiple different values such as 0.0 for no dependence, 0.3 for moderate, and 0.5 for strong dependence, and reporting each.
- σ^2 The residual variance. This should be based on previous similar experiments, or a conservative estimate. Underestimating σ^2 will result in *overstated power*.
- δ Detectable treatment difference. Choose δ as the *smallest* treatment difference that would matter scientifically, agronomically, commercially, etc. So if yield differences of less than 0.1 t/ha are not important to the farmer, then use $\delta = 0.1$. Otherwise, test multiple deltas and report the power of each of them.

Value

A list containing:

contrast_power Data frame of pairwise treatment-comparison standard errors, noncentrality parameters, power values, and effective replicates.

design_power The minimum pairwise power across all estimable treatment comparisons.

average_power The average pairwise power across all estimable treatment comparisons.

worst_comparison The treatment comparison with the lowest power.

fisher_info The treatment information matrix.

treatment_cov The model-based covariance matrix of treatment estimates, equal to $\sigma^2 I_T^+$.

eigenvalues Positive eigenvalues of the treatment information matrix.

rank Numerical rank of the treatment information matrix.

assumptions List of covariance, effect-size, and testing assumptions used.

Examples

```
## Not run:
# RCBD
df <- data.frame(
    row = rep(1:6, each = 4),
    col = rep(1:4, times = 6),
    treatment = rep(LETTERS[1:8], 3),
    block = rep(1:3, each = 8)
)

# Optimise while respecting blocks
result <- speed::speed(df,
    "treatment",
    swap_within = "block",
    iterations = 5000,
    seed = 42
)

## test on latin square
latinsquare <- data.frame(
    row = rep(1:4, each = 4),
    col = rep(1:4, times = 4),
    treatment = c(
        "A", "B", "C", "D",
        "B", "C", "D", "A",
        "C", "D", "A", "B",
        "D", "A", "B", "C"
    )
)

res_power <- design_power(
    design = latinsquare,
    treatment_cols = "treatment",
    row_col = "row",
```

```
    column_col = "col",
    block_col = NULL,
    rho_row = 0.3,
    rho_col = 0.3,
    sigma2 = 1,
    delta = 1,
    alpha = 0.05
)

## End(Not run)
```

`directional_variograms` *A Spatial Variogram Function for Single Site Data*

Description

This function allows you to observe directional variograms in the 0 (vertical) and 90 (horizontal) directions for gridded small-plot trial data.

Usage

```
directional_variograms(model)
```

Arguments

`model`	`asreml` A model with some residual structure.

Value

`gstat::variogram` A **Gstat** variogram.

Examples

```
library(asreml)
model <- asreml(
    fixed = yield ~ weed,
    random = ~idv(Variety),
    residual = ~ar1v(Row):id(Column),
    data = wheat
)
mod <- directional_variograms(model)
plot(mod, multipanel = FALSE)

## Not run:
# You can also plot using ggplot if you wish
ggplot(
    mod,
    aes(x = dist, y = gamma, group = dir.hor, colour = factor(dir.hor))
```

```
) +
    geom_point() +
    geom_line() +
    facet_grid(~dir.hor, scales = "free_x")

## End(Not run)
```

`exploratory_table` *Produces a summary table for the data with Mean, Median, SD, CV and Range*

Description

This function outputs the summary statistics of the given data.

Usage

```
exploratory_table(data, response, groupby)
```

Arguments

`data`	`data.frame` A data set to be summarised
`response`	`character` String variable name in the data of the response
`groupby`	`character` vector Vector of strings of the names of the grouping variables in the data

Value

`data.frame` Summary table

Examples

```
exploratory_table(mtcars, response = "mpg", groupby = c("cyl", "gear"))
```

`lsd_graph` *Create an LSD graph from an ASReml model*

Description

Generates a table of least significant differences (LSDs) for a given model.

Usage

```
lsd_graph(model, classify, ...)
```

Arguments

model	The model to generate LSDs from. The value may be: • asreml • glmmTMB (not yet implemented)
classify	character vector A string specifying which variables to predict and calculate LSDs from. For asreml models, it is passed to classify argument of predictPlus.asreml.
...	Arguments to pass to predictPlus.asreml

Value

ggplot object Returns a ggplot2 plot of the LSDs.

Examples

```
library(asreml)
model <- asreml(
    fixed = yield ~ Variety + Nitrogen + Variety:Nitrogen,
    random = ~idv(Blocks) + idv(Blocks):idv(Wplots),
    residual = ~idv(units),
    data = oats
)
lsd_graph(model, classify = "Variety")
```

---

lsd_group *Group treatments using results of paired t-test*

Description

Intended as a replacement of the agricolae::orderPvalue() function, but with (maybe) a better algorithm.

Usage

```
lsd_group(treatments, means, alpha, pvalues)
```

Arguments

treatments	character vector Character vector of the treatment names
means	numeric Vector of the treatment means/fitted values
alpha	numeric Significant difference threshold
pvalues	matrix of numeric Matrix of pvalues calculated via pairwise t-tests

Value

data.frame Dataframe of each treatment and their associated LSD group

Examples

```
library(asreml)
model <- asreml(
    fixed = yield ~ Variety + Nitrogen + Variety:Nitrogen,
    random = ~idv(Blocks) + idv(Blocks):idv(Wplots),
    residual = ~idv(units),
    data = oats
)

pred <- asremlPlus::predictPlus.asreml(
    model,
    classify = "Variety",
    wald.tab = as.data.frame(asreml::wald(model, denDF = "algebraic")$Wald)
)

prob.matrix <- ifelse(is.na(pred$p.differences), 1, pred$p.differences)
treatments <- colnames(prob.matrix)
means <- pred$predictions$predicted.value
alpha <- 0.05

lsd_group(
    treatments,
    means,
    alpha,
    prob.matrix
)
```

---

lsd_table *Create an LSD table from an ASReml model*

Description

Generates a table of least significant differences (LSDs) for a given model.

Usage

```
lsd_table(model, classify, alpha = 0.05, ...)
```

Arguments

model	The model object to calculate LSDs from. The value may be: • asreml

- `glmmTMB` (not yet implemented)

`classify` `character` A string specifying which variables to predict and calculate LSDs from.

`alpha` significance level `numeric`.

`...` Arguments to pass to `predictPlus.asreml`

Value

A `data.frame` with the LSD values.

Examples

```
library(asreml)
model <- asreml(
    fixed = yield ~ Variety + Nitrogen + Variety:Nitrogen,
    random = ~ idv(Blocks) + idv(Blocks):idv(Wplots),
    residual = ~ idv(units),
    data = oats
)
lsd_table(model, classify = "Variety")
```

---

ofe_grid_data *Grid data ready for ASReml analysis*

Description

Optimally rotate and grid a georeferenced dataframe

Usage

```
ofe_grid_data(data_in, rotation_angle, nrow, npe, ncol, trim_ends = TRUE)
```

Arguments

`data_in` `data.frame`. The dataframe to be rotated. This should be georeferenced per the **sf** package.

We assume there are only three column of actual interest in the dataframe, which are:

Yield The observed yield values.

Treatment The treatment factor to assess.

Rep The blocking structure to be preserved

`rotation_angle` `numeric`. Clockwise angle in degrees to rotate the dataframe.

`nrow` `numeric`. The number of rows in the final grid. This should be as large as possible without introducing spurious NA values.

npe	numeric. Number of pseudo-environments to be generated along the strips. PEs will be evenly spaced, so may not be appropriate in all cases. Always assess PEs visually before use.
ncol	numeric. The actual number of strips in the data. This should be counted/confirmed before hand. If there is a gap between strips, count the number of strips in this gap when determining this number.
trim_ends	bool. Boolean determining whether the gridded data should be trimmed to remove NAs occurring at the ends of columns. Defaults to FALSE.

Value

gridded.ofe. list containing the following items:

gridded_data data.frame The georeferenced gridded data.

data_original data.frame The input data with additional columns for visualising the rotation process.

Author(s)

Braden Thorne, <braden.thorne@curtin.edu.au>

Examples

```
library(CBADASReml)

data <- agridat::lasrosas.corn |>
filter(year == 2001) |>
rename(
  Yield = yield,
  Treatment = nf,
  Rep = rep
 ) |>
 dplyr::select(Yield, Treatment, Rep, long, lat) |>
 st_as_sf(coords = c("long", "lat"), crs = 4326) |>
 st_transform(3395)

ofe_grid_data(data, -60, 80, 5, 18, trim_ends=FALSE)
```

---

ofe_rotate_data *Rotate a georeferenced data frame*

Description

Rotate a provided georeferenced dataframe by a specified angle in degrees.

Usage

```
ofe_rotate_data(data, angle)
```

Arguments

data	data.frame. The dataframe to be rotated. This should be georeferenced per the sf package.
angle	numeric. Clockwise angle in degrees to rotate the dataframe.

Value

data.frame. data with geometry now defined in rotated coordinates. Contains the same columns as the input with the following additions:

x_original The original x-coordinates before rotation.

y_original The original y-coordinates before rotation.

Author(s)

Braden Thorne, <braden.thorne@curtin.edu.au>

Examples

```
library(CBADASReml)

data <- agridat::lasrosas.corn |>
filter(year == 2001) |>
rename(
  Yield = yield,
  Treatment = nf,
  Rep = rep
 ) |>
 dplyr::select(Yield, Treatment, Rep, long, lat) |>
 st_as_sf(coords = c("long", "lat"), crs = 4326) |>
 st_transform(3395)

ofe_rotate_data(data, -60)
```

---

outlier_summary *Detect outliers for small-plot trial analysis*

Description

Provides a summary of the outliers present in the asreml model. Gives context to the outliers by showing the responses for the same factor combinations as the outliers

Usage

```
outlier_summary(model, cutoff = 3.5)
```

Arguments

model	The model object to detect outliers in. The value may be: • asreml • glmmTMB (not yet implemented)
cutoff	numeric The magnitude of a point's residual to be considered an outlier.

Value

NULL Prints:

- The number of outliers.
- A table of the outliers, if any.
- A table of relevant context to some factor combinations, if there are outliers.

Examples

```
library(asreml)
model <- asreml(
    fixed = weight ~ littersize + Dose + Sex + Dose:Sex,
    random = ~ idv(Dam),
    residual = ~units,
    data = rats
)
outlier_summary(model)
```

---

plot_gridded_ofe *Plot diagnostics of gridded OFE*

Description

Returns a plot with the following attributes:

Original Data TOP LEFT. The raw data in the projected space.

Rough Rotation TOP RIGHT Rotated original data with alternative strips highlighted. If each strip is not a single colour, the rotation is insufficient. In practice only the first strip needs to be identified correctly, however it is best to ensure all strips are identified.

Gridded Data (Raw) BOTTOM LEFT The accurate rotated, gridded data. Strips should be vertically aligned. NAs will display in red. The number of NAs should be relatively small, concentrated around the edges where strips do not begin at the same point (if ends were trimmed these should not be present) and in holes where data is missing. Red horizontal lines indicate that too many rows have been specified. Large NA concentrations in all four corners indicates the rotation has likely failed.

Gridded Data (Projected) BOTTOM RIGHT The gridded data projected back to the original data space. This should look similar to the original data - if it does not the process has likely failed and the other plots should be inspected to establish why.

Usage

```
plot_gridded_ofe(gridded_ofe)
```

Arguments

gridded_ofe	gridded.ofe. list generated by ofe_grid_data containing the following items: **gridded_data** data.frame The georeferenced gridded data. **data_original** data.frame The input data with additional columns for visualising the rotation process.

Author(s)

Braden Thorne, <braden.thorne@curtin.edu.au>

Examples

```
library(CBADASReml)

data <- agridat::lasrosas.corn |>
filter(year == 2001) |>
rename(
  Yield = yield,
  Treatment = nf,
  Rep = rep
 ) |>
 dplyr::select(Yield, Treatment, Rep, long, lat) |>
 st_as_sf(coords = c("long", "lat"), crs = 4326) |>
 st_transform(3395)

gridded_data <- ofe_grid_data(data, -60, 80, 5, 18, trim_ends=FALSE)
plot_gridded_ofe(gridded_data)
```

---

pred_table *Generate prediction table from model*

Description

Generate a table of predicted values and CIs from an asreml or glmmTMB model, for the specified variables asreml or glmmTMB object.

Usage

```
pred_table(
  mod,
  classify,
  link_fun = "identity",
```

```
  tmb_component = "cond",
  tmb_type = "response",
  factor_combine = TRUE,
  trt_col_label = "Treatment"
)
```

Arguments

mod	The model object to get predictions from. The value may be: • asreml • glmmTBM
classify	character vector. Variable for which predictions will be made. For asreml::asreml() models, it is passed to classify argument of asremlPlus::predictPlus.asrer For **glmmTMB** models it is used in the specs argument of emmeans::emmeans()
link_fun	character vector. Specifies the transformation function to apply over the predictions. Only applies for asreml::asreml() objects. The value may be: • "identity" (default) • "log" • "inverse" • "sqrt" • "logit" • "probit" • "cloglog"
tmb_component	character vector. Specifies the component of the **glmmTMB** model from which to get predictions. The value may be: • "cond" (default) • "zi" • "cmean" • "response"
tmb_type	character vector. Specifies the prediction type for **glmmTMB** models. • Only "response" is supported at the moment.
factor_combine	Logical Whether or not to combine the factors in classify into a single column in the output table. If TRUE (default), the factor levels are concatenated and labelled using trt_col_label.
trt_col_label	character vector. Specifies the label for the combined factor column when factor_combine is TRUE. Defaults to "Treatment"

Value

data.frame. Contains the predicted means, standard errors, and confidence intervals for the specified variables. Includes the following columns:

Treatment The combined factor levels, if factor_combine is TRUE.

Mean The predicted mean values.

Standard Error The standard errors of the predicted means.

Lower CL The lower confidence limits.

Upper CL The upper confidence limits.

If factor_combine is FALSE, the factors specified in classify remain as separate columns.

Author(s)

Matthew Nguyen, <matthew.nguyen@curtin.edu.au>

Examples

```
library(CBADASReml)
library(asreml)
library(glmmTMB)
mod1 <- asreml(
    fixed = yield ~ Variety + Nitrogen + Variety:Nitrogen,
    random = ~ idv(Blocks) + idv(Blocks):idv(Wplots),
    residual = ~ idv(units),
    data = oats
)
# Zero inflated model
mod2 <- glmmTMB(
    count ~ spp * mined + (1|site),
    zi = ~ spp * mined,
    data = Salamanders,
    family = nbinom2
)
pred_table(mod1, classify = "Variety")
```

---

report_tables *A prediction function for models with more than one variable to predict on*

Description

This function outputs the predictions for each factor for a single model. Each factor is put on a separate Excel sheet.

Usage

```
report_tables(model, classify)
```

Arguments

model	The model to generate predictions for. The value may be: • asreml • glmmTBM (not yet implemented)
classify	character A string specifying the factors in the model to predict on. If multiple are specified separate with either * or :. For example Nitrogen:Variety or Nitrogen*Variety.

Value

list of data.frame A list of data frames. The first data frame is the ANOVA for the model. The remaining data frames are the prediction tables from the classify object.

Examples

```
library(asreml)
model <- asreml(
    fixed = yield ~ Variety + Nitrogen + Variety:Nitrogen,
    random = ~idv(Blocks) + idv(Blocks):idv(Wplots),
    residual = ~idv(units),
    data = oats
)

report_tables(
    model,
    classify = "Variety:Nitrogen"
)

## Not run:
# Using it to write with writexl
tables <- excel_prediction_file(
    model,
    classify = "Variety:Nitrogen",
)

writexl::write_xlsx(x = tables, path = "Prediction_Tables.xlsx")

## End(Not run)
```

Index