Package: agriutilities (via r-universe)

September 9, 2024

Type Package

Title Utilities for Data Analysis in Agriculture

Version 1.2.0.9000

Description Utilities designed to make the analysis of field trials easier and more accessible for everyone working in plant breeding. It provides a simple and intuitive interface for conducting single and multi-environmental trial analysis, with minimal coding required. Whether you're a beginner or an experienced user, 'agriutilities' will help you quickly and easily carry out complex analyses with confidence. With built-in functions for fitting Linear Mixed Models, 'agriutilities' is the ideal choice for anyone who wants to save time and focus on interpreting their results. Some of the functions require the R package 'asreml' for the 'ASReml' software, this can be obtained upon purchase from 'VSN' international (<https://vsni.co.uk/software/asreml-r>).

License MIT + file LICENSE

Imports ggplot2, psych, dplyr, tidyr, lme4, Matrix, ggpubr, lmerTest, data.table, stats, magrittr, emmeans, ggrepel, tibble, rlang, statgenSTA, SpATS

Enhances asreml

Encoding UTF-8

RoxygenNote 7.3.1

URL https://github.com/AparicioJohan/agriutilities, https://apariciojohan.github.io/agriutilities/

BugReports https://github.com/AparicioJohan/agriutilities/issues

ByteCompile TRUE

Suggests knitr, lattice, cluster, rmarkdown, agridat, gt

VignetteBuilder knitr

Repository https://apariciojohan.r-universe.dev

RemoteUrl https://github.com/apariciojohan/agriutilities

RemoteRef HEAD

RemoteSha 59feb146a0319534fa455f7c3041c07235b1df24

Contents

`check_connectivity` *Check connectivity between trials*

Description

Check connectivity between trials

Usage

```
check_connectivity(
  data = NULL,
  genotype = "line",
  trial = "Experiment",
  response = NULL,
  all = FALSE,
  return_matrix = FALSE
)
```

Arguments

data	A data.frame in a wide format.
genotype	A character string indicating the column in data that contains genotypes.
trial	A character string indicating the column in data that contains trials.
response	A character string specifying the trait.
all	Whether or not print all the table.
return_matrix	A logical value indicating if the user wants to return a (n_trial x n_trial) matrix with the amount of genotypes shared between each pair of trial. (FALSE by default)

Value

A data.frame with the genotype connectivity. If return_matrix is TRUE, it will return a n_trial x n_trial matrix with the amount of genotypes shared between each pair of trial.

Examples

```
library(agridat)
library(agriutilities)
data(besag.met)
dat <- besag.met
head(
  check_connectivity(
    data = dat,
    genotype = "gen",
    trial = "county",
    response = "yield",
    all = TRUE,
    return_matrix = FALSE
  )
)
```

---

check_design_met *Check Experimental Design*

Description

This function helps to identify the experimental design of each trial, filters the data and then provide a summary for the traits and the experimental design. This works as a quality check before we fit any model. Returns an object of class checkAgri.

Usage

```
check_design_met(
  data = NULL,
  genotype = NULL,
  trial = NULL,
  traits = NULL,
  rep = NULL,
  block = NULL,
  row = NULL,
  col = NULL
)
```

Arguments

data	A data.frame in a wide format.
genotype	A character string indicating the column in data that contains genotypes.
trial	A character string indicating the column in data that contains trials.
traits	A character vector specifying the traits for which the models should be fitted.
rep	A character string indicating the column in data that contains replicates.
block	A character string indicating the column in data that contains sub blocks.
row	A character string indicating the column in data that contains the row coordinates.
col	A character string indicating the column in data that contains the column coordinates.

Value

An object of class `checkAgri`, with a list of:

summ_traits	A data.frame containing a summary of the traits.
exp_design_resum	A data.frame containing a summary of the experimental design.
filter	A list by trait containing the filtered trials.
exp_design_list	A data.frame containing the experimental design of each trial.
check_connectivity	A data.frame with the genotype connectivity.
connectivity_matrix	A matrix with the amount of genotypes shared between each pair of trial.
data_design	A data frame containing the data used with two additional columns, one realted to the experimental design and a sequential number (id)
inputs	A list containing the character string that indicates the column in data that contains the genotype, trial, traits, rep, block, row and col.

Examples

```
library(agridat)
library(agriutilities)
data(besag.met)
dat <- besag.met
results <- check_design_met(
  data = dat,
  genotype = "gen",
  trial = "county",
  traits = c("yield"),
  rep = "rep",
  block = "block",
  col = "col",
  row = "row"
)
print(results)
plot(results, type = "connectivity")
plot(results, type = "missing")
```

`covcor_heat` *Correlation Covariance Heatmap*

Description

Correlation Covariance Heatmap

Usage

```
covcor_heat(
  matrix,
  corr = TRUE,
  size = 4,
  digits = 3,
  legend = c(0.6, 0.7),
  upper_tri = FALSE,
  reorder = FALSE
)
```

Arguments

`matrix`	A numeric matrix.
`corr`	A logical value indicating if the matrix is in a scaled form (`TRUE` by default, correlation matrix)
`size`	A numeric value to define the letter size.
`digits`	A numeric integer to define the number of digits to plot.
`legend`	the position of legends ("none", "left", "right", "bottom", "top", or two-element numeric vector)

upper_tri	A logical value to plot the Lower or Upper Triangular Part of the matrix. FALSE by default.
reorder	A logical value to reorder by a Hierarchical Clustering. FALSE by default.

Value

A ggplot object showing the upper triangular elements of the matrix.

Examples

```
library(agriutilities)
data(iris)
M <- cor(iris[, -5])
covcor_heat(matrix = M, corr = TRUE)
```

---

extract_rcov *Extract Residual Variance-Covariance from ASReml-R*

Description

This function is specially useful for extracting residual variance covariance matrices from ASReml-R when running repeated measurements analysis.

Usage

```
extract_rcov(model = NULL, time = NULL, plot = NULL, vc_error = NULL)
```

Arguments

model	An asreml object.
time	An optional character string indicating the "Time". By default the function identifies this parameter.
plot	An optional character string indicating the "PlotID". By default the function identifies this parameter.
vc_error	An optional character string indicating the variance covariance. It can be "corv", "corh", "corgh", "us", "expv", "exph", "ar1v", "ar1h" or "ante". By using NULL the function tries to guess which was the variance-covariance used.

Details

The expected residual variance covariance structure must be of the form: '~id(Plot):corv(Time)', where 'Plot' is a unique identifier of each experimental unit, and 'Time', represents the variable that contains the time when the experimental units were measured. This form also requires that the levels of the factor Time are nested in the levels of the factor Plot. If it is not in that form you can sort the dataset by using the following command 'arrange(grassUV, Plant, Time)'.

Value

An object with a list of:

corr_mat	A matrix with the residual correlation between time points.
vcov_mat	A matrix of the estimated residual variance-covariance between time points.
vc	A character string indicating the variance-covariance fitted.

Examples

```
## Not run:
library(ggpubr)
library(agriutilities)
library(tidyverse)
library(asreml)

head(grassUV)
str(grassUV)

# Exploration ------------------------------------------------------------

grassUV %>%
  ggplot(
    aes(x = Time, y = y, group = Plant, color = Plant)
  ) +
  geom_point() +
  geom_line() +
  facet_wrap(~Tmt) +
  theme_minimal(base_size = 15)

tmp <- grassUV %>%
  group_by(Time, Plant) %>%
  summarise(mean = mean(y, na.rm = TRUE)) %>%
  spread(Time, mean) %>%
  column_to_rownames("Plant")

gg_cor(tmp, label_size = 5)

tmp %>%
  cor(use = "pairwise.complete.obs") %>%
  as.data.frame() %>%
  rownames_to_column(var = "Time") %>%
  gather("DAP2", "corr", -1) %>%
  type.convert(as.is = FALSE) %>%
  mutate(corr = ifelse(Time < DAP2, NA, corr)) %>%
  mutate(DAP2 = as.factor(DAP2)) %>%
  ggplot(
    aes(x = Time, y = corr, group = DAP2, color = DAP2)
  ) +
  geom_point() +
  geom_line() +
  theme_minimal(base_size = 15) +
  color_palette(palette = "jco") +
```

```
  labs(color = "Time", y = "Pearson Correlation")

# Modeling ----------------------------------------------------------------

# Identity variance model.
model_0 <- asreml(
  fixed = y ~ Time + Tmt + Tmt:Time,
  residual = ~ id(Plant):idv(Time),
  data = grassUV
)

# Simple correlation model; homogeneous variance form.
model_1 <- asreml(
  fixed = y ~ Time + Tmt + Tmt:Time,
  residual = ~ id(Plant):corv(Time),
  data = grassUV
)

# Exponential (or power) model; homogeneous variance form.
model_2 <- asreml(
  fixed = y ~ Time + Tmt + Tmt:Time,
  residual = ~ id(Plant):expv(Time),
  data = grassUV
)

# Exponential (or power) model; heterogeneous variance form.
model_3 <- asreml(
  fixed = y ~ Time + Tmt + Tmt:Time,
  residual = ~ id(Plant):exph(Time),
  data = grassUV
)

# Antedependence variance model of order 1
model_4 <- asreml(
  fixed = y ~ Time + Tmt + Tmt:Time,
  residual = ~ id(Plant):ante(Time),
  data = grassUV
)

# Autoregressive model of order 1; homogeneous variance form.
model_5 <- asreml(
  fixed = y ~ Time + Tmt + Tmt:Time,
  residual = ~ id(Plant):ar1v(Time),
  data = grassUV
)

# Autoregressive model of order 1; heterogeneous variance form.
model_6 <- asreml(
  fixed = y ~ Time + Tmt + Tmt:Time,
  residual = ~ id(Plant):ar1h(Time),
  data = grassUV
)
```

```
# Unstructured variance model.
model_7 <- asreml(
  fixed = y ~ Time + Tmt + Tmt:Time,
  residual = ~ id(Plant):us(Time),
  data = grassUV
)

# Model Comparison ----------------------------------------------------

models <- list(
  "id" = model_0,
  "cor" = model_1,
  "exp" = model_2,
  "exph" = model_3,
  "ante" = model_4,
  "ar1" = model_5,
  "ar1h" = model_6,
  "us" = model_7
)

summary_models <- data.frame(
  model = names(models),
  aic = unlist(lapply(models, \(x) summary(x)$aic)),
  bic = unlist(lapply(models, \(x) summary(x)$bic)),
  loglik = unlist(lapply(models, \(x) summary(x)$loglik)),
  nedf = unlist(lapply(models, \(x) summary(x)$nedf)),
  param = unlist(lapply(models, \(x) attr(summary(x)$aic, "param"))),
  row.names = NULL
)

summary_models %>%
  ggplot(
    aes(x = reorder(model, -bic), y = bic, group = 1)
  ) +
  geom_point(size = 2) +
  geom_text(aes(x = model, y = bic + 5, label = param)) +
  geom_line() +
  theme_minimal(base_size = 15) +
  labs(x = NULL)

# Extracting Variance Covariance Matrix -----------------------------------

extract_rcov(model_4)

covcor_heat(
  matrix = extract_rcov(model_1)$corr,
  legend = "none",
  size = 5
) + ggtitle(label = "Uniform Correlation (corv)")
covcor_heat(
  matrix = extract_rcov(model_2)$corr,
  legend = "none",
  size = 5
```

```
) + ggtitle(label = "Exponetial (expv)")

## End(Not run)
```

`extract_vcov` *Extract Variance-Covariance from ASReml-R*

Description

Extract Variance-Covariance from ASReml-R

Usage

```
extract_vcov(model = NULL, gen = "genotype", env = "trial", vc_model = "corv")
```

Arguments

`model`	ASReml object
`gen`	A character string indicating the column in data that contains genotypes.
`env`	A character string indicating the column in data that contains environments or trials.
`vc_model`	A character string indicating the variance-covariance fitted. Can be 'diag', 'corv', 'corh', 'corgv', 'fa1', 'fa2', 'fa3', 'fa4', 'corgh', 'us' or 'rr2'.

Value

An object with a list of:

`VCOV`	A matrix of the estimated variance-covariance between trials.
`CORR`	A n_trial x n_trial matrix with the correlation between trials.
`vc_model`	A character string indicating the variance-covariance fitted.

Examples

```
## Not run:
library(agridat)
library(agriutilities)

data(besag.met)
dat <- besag.met
results <- check_design_met(
  data = dat,
  genotype = "gen",
  trial = "county",
  traits = c("yield"),
  rep = "rep",
  block = "block",
  col = "col",
```

```
  row = "row"
)
out <- single_trial_analysis(results, progress = FALSE)
met_results <- met_analysis(out, progress = FALSE)
extract_vcov(
  model = met_results$met_models$yield,
  vc_model = "us"
)

## End(Not run)
```

`fa_summary` *Factor Analytic Summary*

Description

Factor Analytic Summary

Usage

```
fa_summary(
  model = NULL,
  trial = "trial",
  genotype = "genotype",
  BLUEs_trial = NULL,
  mult_fa1 = -1,
  mult_fa2 = 1,
  filter_score = 1.5,
  k_biplot = 1,
  size_label_var = 2,
  alpha_label_var = 0.2,
  size_label_ind = 2,
  alpha_label_ind = 0.8,
  size_arrow = 0.2,
  alpha_arrow = 0.2,
  base_size = 12
)
```

Arguments

`model`	Factor Analytic Model (ASReml object)
`trial`	A character string indicating the column in data that contains trials.
`genotype`	A character string indicating the column in data that contains genotypes.
`BLUEs_trial`	A data.frame containing BLUEs for each trial.
`mult_fa1`	A constant to multiply the first loading. Must be 1 or -1. (-1 by default)
`mult_fa2`	A constant to multiply the second loading. Must be 1 or -1. (1 by default)

filter_score	A numeric value to filter genotypes by the distance from the origin.
k_biplot	A numeric value to multiply the scores in the biplot.
size_label_var	A numeric value to define the label size for the variables.
alpha_label_var	A numeric value between (0,1) to define the label for the variables.
size_label_ind	A numeric value to define the label size for the individuals.
alpha_label_ind	A numeric value between (0,1) to define the label for the individuals.
size_arrow	A numeric value to define the arrow size.
alpha_arrow	A numeric value between (0,1) to define the arrow.
base_size	A numeric value to define the base size.

Value

An object with a list of:

loadings	A data.frame containing the first and second loading for each trial.
loading_star	A data.frame containing the first and second loading rotated for each trial.
Gvar	A matrix of the estimated variance-covariance between trials.
Cmat	A matrix of the correlation between trials.
summary_loading	A data.frame containing a summary of the loadings.
paf_site	A data.frame containing the percentage of variance explained for each component and for each trial.
var_tot	A numeric value of the total variance.
scores	A data.frame containing the scores for each genotype.
plots	A list with different plots. Includes a plot for the loadings, biplot, biplot_scaled and loadings_c.

Examples

```
## Not run:
library(agridat)
library(agriutilities)
data(besag.met)
dat <- besag.met
results <- check_design_met(
  data = dat,
  genotype = "gen",
  trial = "county",
  traits = c("yield"),
  rep = "rep",
  block = "block",
  col = "col",
  row = "row"
)
```

```
out <- single_trial_analysis(results, progress = FALSE)
met_results <- met_analysis(out, vcov = "fa2", progress = FALSE)

pp <- met_results$trial_effects
model <- met_results$met_models$yield
fa_summary(
  model = model,
  trial = "trial",
  genotype = "genotype",
  BLUEs_trial = pp,
  mult_fa1 = -1,
  mult_fa2 = -1,
  filter_score = 1,
  k_biplot = 10,
  size_label_var = 3,
  alpha_label_var = 0.5,
  size_label_ind = 3,
  alpha_label_ind = 0.8,
  size_arrow = 0.2,
  alpha_arrow = 0.1
)

## End(Not run)
```

---

gg_cor *Triangular Correlation Plot*

Description

Return a ggplot object to plot a triangular correlation figure between 2 or more variables.

Usage

```
gg_cor(
  data,
  colours = c("#db4437", "white", "#4285f4"),
  blackLabs = c(-0.7, 0.7),
  show_signif = TRUE,
  p_breaks = c(0, 0.001, 0.01, 0.05, Inf),
  p_labels = c("***", "**", "*", "ns"),
  show_diagonal = FALSE,
  diag = NULL,
  return_table = FALSE,
  return_n = FALSE,
  adjusted = TRUE,
  label_size = 3,
  method = "pearson"
)
```

Arguments

data	A data.frame with numerical columns for each variable to be compared.
colours	A vector of size three with the colors to be used for values -1, 0 and 1.
blackLabs	A numeric vector of size two, with min and max correlation coefficient.
show_signif	Logical scalar. Display significance values ?
p_breaks	Passed to function 'cut'. Either a numeric vector of two or more unique cut points or a single number (greater than or equal to 2) giving the number of intervals into which x is to be cut.
p_labels	Passed to function 'cut'. labels for the levels of the resulting category. By default, labels are constructed using "(a,b]" interval notation. If p_labels = FALSE, simple integer codes are returned instead of a factor.
show_diagonal	Logical scalar. Display main diagonal values ?
diag	A named vector of labels to display in the main diagonal. The names are used to place each value in the corresponding coordinates of the diagonal. Hence, these names must be the same as the colnames of data.
return_table	Return the table to display instead of a ggplot object.
return_n	Return plot with shared information.
adjusted	Use the adjusted p values for multiple testing instead of raw coeffs. TRUE by default.
label_size	Numeric value indicating the label size. 3 by default.
method	method="pearson" is the default value. The alternatives to be passed to cor are "spearman" and "kendall". These last two are much slower, particularly for big data sets.

Value

A ggplot object containing a triangular correlation figure with all numeric variables in data. If return_table is TRUE, the table used to produce the figure is returned instead.

Author(s)

Daniel Ariza, Johan Aparicio.

Examples

```
library(agriutilities)
data(iris)
gg_cor(
  data = iris,
  colours = c("#db4437", "white", "#4285f4"),
  label_size = 6
)
```

heritability_fa *Heritability for Factor Analytic Models in ASReml-R*

Description

Heritability for Factor Analytic Models in ASReml-R

Usage

```
heritability_fa(
  model_fa = NULL,
  genotype = "line",
  env = "loc",
  vc_model = c("fa2"),
  diag = FALSE
)
```

Arguments

model_fa	Factor Analytic ASReml model
genotype	A character string indicating the column in data that contains genotypes.
env	A character string indicating the column in data that contains environments or trials.
vc_model	A character string indicating the variance-covariance structure. Can be "fa1", "fa2", "fa3", "fa4" or "us".
diag	TRUE or FALSE depending on the user if they want to take the elements on the diagonal of the variance-covariance matrix or the elements out of the diagonal to estimate the heritability. FALSE by default.

Value

An object with a list of:

h2_cullis	A numerical value of the Cullis heritability estimate.
h2_se	A numerical value of the Cullis heritability estimate based on the standard error.

Examples

```
## Not run:
library(agridat)
library(agriutilities)
data(besag.met)
dat <- besag.met
results <- check_design_met(
  data = dat,
  genotype = "gen",
  trial = "county",
```

```
  traits = c("yield"),
  rep = "rep",
  block = "block",
  col = "col",
  row = "row"
)
out <- single_trial_analysis(results, progress = FALSE)
met_results <- met_analysis(out, progress = FALSE)
model <- met_results$met_models$yield
heritability_fa(
   model_fa = model,
   genotype = "genotype",
   env = "trial",
   vc_model = "us"
 )

## End(Not run)
```

`h_cullis` *Cullis heritability for lme4 models*

Description

Cullis heritability for lme4 models

Usage

```
h_cullis(model, genotype, re_MME = FALSE)
```

Arguments

`model`	Object of class `lmer`.
`genotype`	A character string indicating the column in data that contains genotypes.
`re_MME`	A logical value to ask if we want to reconstruct the mixed models equations to estimate the Cullis heritability. (`FALSE` by default)

Value

A numerical value of the Cullis heritability estimate. If `re_MME` is `TRUE`, a list with matrices of the mixed models equations is returned.

Author(s)

Paul Schmidt, Johan Aparicio.

Examples

```
library(lme4)
library(agridat)
library(agriutilities)
dat <- john.alpha
g.ran <- lmer(
  formula = yield ~ rep + (1 | gen) + (1 | rep:block),
  data = dat
)
h_cullis(model = g.ran, genotype = "gen")
```

---

h_cullis_spt *Calculate Cullis heritabilities from SpATS objects*

Description

Calculate Cullis heritabilities from SpATS objects

Usage

```
h_cullis_spt(model)
```

Arguments

model	an object of class SpATS as produced by SpATS()

Value

A data frame. The data frame has the following components

- trait : Character string with the trait being analyzed
- H2Cullis : Generalized heritability proposed by Cullis (2006)
- H2Oakey : Generalized heritability proposed by Oakey (2006)
- reBLUP_avg : Average BLUP reliability
- vdBLUP_avg : Average pairwise prediction error variance of genotype effects
- PEV_avg : Average predictive error variance (PEV) of genotype effects
- var_G : Genotypic Variance

Author(s)

Johan Aparicio

References

Cullis, B. R., Smith, A. B., & Coombes, N. E. (2006). On the design of early generation variety trials with correlated data. Journal of agricultural, biological, and environmental statistics, 11, 381-393.

Oakey, H., A. Verbyla, W. Pitchford, B. Cullis, and H. Kuchel (2006). Joint modeling of additive and non-additive genetic line effects in single field trials. Theoretical and Applied Genetics, 113, 809 - 819.

Examples

```
library(SpATS)
library(agriutilities)
data(wheatdata)
wheatdata$R <- as.factor(wheatdata$row)
wheatdata$C <- as.factor(wheatdata$col)

m1 <- SpATS(
  response = "yield",
  spatial = ~ PSANOVA(col, row, nseg = c(10, 20), nest.div = 2),
  genotype = "geno",
  genotype.as.random = TRUE,
  fixed = ~ colcode + rowcode,
  random = ~ R + C,
  data = wheatdata,
  control = list(tolerance = 1e-03, monitoring = 0)
)

h_cullis_spt(m1)
```

---

ic_reml_asr *Find the AIC and BIC for a set of models fitted using asreml*

Description

The function is a wrapper for 'icREML' function described in Verbyla (2019).

Usage

```
ic_reml_asr(fm, scale = 1)
```

Arguments

fm	A list of asreml fitted model objects
scale	A scalar to scale the variance matrix of the estimated fixed effects (to ensure numerical stability of a log-determinant). Default value is 1.

Value

A data frame. The data frame has the following components

- `model` : the names of the models
- `loglik` : the full log-likelihood for each model
- `p` : the number of fixed effects parameters for each model
- `q` : the number of (non-zero) variance parameters for each model.
- `b` : the number of variance parameters that are fixed or on the boundary. These parameters are not counted in the AIC or BIC.
- `AIC` : the AIC for each model
- `BIC` : the BIC for each model
- `logdet` : the log-determinant used in adjusting the residual log-likelihood for each model

Author(s)

Ari Verbyla (averbyla at avdataanalytics.com.au)

References

Verbyla, A. P. (2019). A note on model selection using information criteria for general linear models estimated using REML. Australian & New Zealand Journal of Statistics, 61(1), 39-50.

---

ic_reml_spt *Find the AIC and BIC for a model fitted using SpATS*

Description

This function calculates the AIC and BIC for a model fitted in SpATS following the methodology proposed by Verbyla (2019).

Usage

```
ic_reml_spt(model, scale = 1, k = 2, label = "spats")
```

Arguments

`model`	A model fitted using SpATS.
`scale`	A scalar to scale the variance matrix of the estimated fixed effects (to ensure numerical stability of a log-determinant). Default value is 1.
`k`	An integer value to round ratios when identifying boundary variance parameters. Default value is 2.
`label`	A string to label the model. Default value is "spats".

Value

A data frame. The data frame has the following components

- `model` : the name of the models
- `loglik` : the full log-likelihood for each model
- `p` : the number of fixed effects parameters for each model
- `q` : the number of (non-zero) variance parameters for each model.
- `b` : the number of variance parameters that are fixed or on the boundary. These parameters are not counted in the AIC or BIC.
- `AIC` : the AIC for each model
- `BIC` : the BIC for each model
- `logdet` : the log-determinant used in adjusting the residual log-likelihood for each model

Author(s)

Johan Aparicio

References

Verbyla, A. P. (2019). A note on model selection using information criteria for general linear models estimated using REML. Australian & New Zealand Journal of Statistics, 61(1), 39-50.

Examples

```
library(SpATS)
library(agriutilities)
data(wheatdata)
wheatdata$R <- as.factor(wheatdata$row)
wheatdata$C <- as.factor(wheatdata$col)

m1 <- SpATS(
  response = "yield",
  spatial = ~ PSANOVA(col, row, nseg = c(10, 20), nest.div = 2),
  genotype = "geno",
  genotype.as.random = TRUE,
  fixed = ~ colcode + rowcode,
  random = ~ R + C,
  data = wheatdata,
  control = list(tolerance = 1e-03, monitoring = 0)
)

m2 <- SpATS(
  response = "yield",
  spatial = ~ PSANOVA(col, row, nseg = c(10, 20), nest.div = 2),
  genotype = "geno",
  genotype.as.random = TRUE,
  fixed = ~colcode,
  random = ~ R + C,
  data = wheatdata,
```

```
  control = list(tolerance = 1e-03, monitoring = 0)
)

rbind.data.frame(
  ic_reml_spt(m1, label = "colcode_rowcode"),
  ic_reml_spt(m2, label = "colcode_no_rowcode")
)

rbind.data.frame(
  h_cullis_spt(m1),
  h_cullis_spt(m2)
)
```

`met_analysis` *Multi-Environmental Trial Analysis*

Description

The results of the `single_trial_analysis()` function are used in `met_analysis()` to fit multi-environmental trial models. Returns an object of class `metAgri`, with a list of trial effects, BLUPs, heritability, variance components, stability and the models fitted.

Usage

```
met_analysis(
  sma_output = NULL,
  h2_filter = 0.2,
  workspace = "1gb",
  vcov = NULL,
  filter_traits = NULL,
  remove_trials = NULL,
  progress = TRUE
)
```

Arguments

`sma_output`	Object of class `smaAgri` resulting of executing `single_trial_analysis()` function.
`h2_filter`	Numeric value to filter trials with poor heritability. 0.2 by default.
`workspace`	Sets the workspace for the core `REML` routines in the form of a number optionally followed directly by a valid measurement unit. "128mb" by default.
`vcov`	A character string specifying the Variance-Covariance structure to be fitted. Can be "fa2", "fa1", "us", "corh" or "corv". If `NULL` the function will try to fit an "us" Variance-Covariance and if it fails, it will try with "fa2" and then with "fa1".
`filter_traits`	A character vector with traits to filter. `NULL` by default.
`remove_trials`	A character vector with trials to remove. `NULL` by default.

progress	Should the progress of the modeling be printed. If TRUE, for every trait a line is output indicating that the model is being fitted.

Value

An object of class metAgri, with a list of:

trial_effects	A data.frame containing Trial BLUEs.
overall_BLUPs	A data.frame containing Genotypic BLUPs across trials, by trait.
BLUPs_GxE	A data.frame containing Genotypic BLUPs by trial/trait.
VCOV	A list by trait contanining the variance-covariance fitted.
stability	A data.frame containing several Stability coefficients resulting of executing the function stability().
heritability	A data.frame containing overall heritabilities by trait.
met_models	A list by trait containing the fitted models.

Examples

```
## Not run:
library(agridat)
library(agriutilities)
data(besag.met)
dat <- besag.met
results <- check_design_met(
  data = dat,
  genotype = "gen",
  trial = "county",
  traits = c("yield"),
  rep = "rep",
  block = "block",
  col = "col",
  row = "row"
)
out <- single_trial_analysis(results, progress = FALSE)
met_results <- met_analysis(out, progress = FALSE)
print(met_results)
covcor_heat(matrix = met_results$VCOV$yield$CORR)

## End(Not run)
```

---

parameters_gg *Genetic Gain Parameters*

Description

Genetic Gain Parameters

Usage

```
parameters_gg(model, trait = "trait")
```

Arguments

model	Linear regression model (lm object)
trait	A character string indicating the column in data that contains trials.

Value

A data.frame with some parameters from the linear regression (Slope, se_Slope, Intercept, r2, Pr(>F)) and the percentage of Genetic Gain.

Examples

```
library(ggplot2)
library(agridat)
library(magrittr)
library(agriutilities)

data(baker.barley.uniformity)
dat <- baker.barley.uniformity
head(dat)

model <- lm(yield ~ year, dat)

dat %>%
  na.omit() %>%
  ggplot(
    aes(x = year, y = yield)
  ) +
  geom_point() +
  geom_smooth(method = "lm") +
  theme_bw()

parameters_gg(model = model, trait = "yield")
```

---

plot.checkAgri *Plot an object of class* checkAgri

Description

Create several plots for an object of class checkAgri

Usage

```
## S3 method for class 'checkAgri'
plot(
  x,
  type = c("connectivity", "missing", "boxplot"),
  axis_size = 15,
  text_size = 5,
  ...
)
```

Arguments

x	An object inheriting from class checkAgri resulting of executing the function check_design_met()
type	A character string specifiying the type of plot. "connectivity", "missing" or "boxplot".
axis_size	Numeric input to define the axis size.
text_size	Numeric input to define the text size.
...	Further graphical parameters. For future improvements.

Value

A ggplot object.

Author(s)

Johan Aparicio [aut]

Examples

```
library(agridat)
library(agriutilities)
data(besag.met)
dat <- besag.met
results <- check_design_met(
  data = dat,
  genotype = "gen",
  trial = "county",
  traits = c("yield"),
  rep = "rep",
  block = "block",
  col = "col",
  row = "row"
)
plot(results, type = "missing")
plot(results, type = "boxplot")
```

plot.metAgri *Plot an object of class* metAgri

Description

Create several plots for an object of class metAgri

Usage

```
## S3 method for class 'metAgri'
plot(
  x,
  type = c("correlation", "covariance", "multi_traits"),
  filter_traits = NULL,
  text_size = 4,
  ...
)
```

Arguments

x	An object inheriting from class metAgri resulting of executing the function met_analysis()
type	A character string specifying the type of plot. "correlation", "covariance" or "multi_traits"
filter_traits	An optional character vector to filter traits.
text_size	Numeric input to define the text size.
...	Further graphical parameters passed to covcor_heat().

Value

A ggplot object.

Author(s)

Johan Aparicio [aut]

Examples

```
## Not run:
library(agridat)
library(agriutilities)
data(besag.met)
dat <- besag.met
results <- check_design_met(
  data = dat,
  genotype = "gen",
  trial = "county",
```

```
  traits = c("yield"),
  rep = "rep",
  block = "block",
  col = "col",
  row = "row"
)
out <- single_trial_analysis(results, progress = FALSE)
met_results <- met_analysis(out, progress = FALSE)
print(met_results)
plot(met_results, type = "correlation")
plot(met_results, type = "covariance")

## End(Not run)
```

`plot.smaAgri` *Plot an object of class* `smaAgri`

Description

Create several plots for an object of class `smaAgri`

Usage

```
## S3 method for class 'smaAgri'
plot(
  x,
  type = c("summary", "correlation", "spatial"),
  filter_traits = NULL,
  nudge_y_cv = 3,
  nudge_y_h2 = 0.07,
  horizontal = FALSE,
  theme_size = 15,
  axis_size = 8,
  text_size = 4,
  ...
)
```

Arguments

`x`	An object inheriting from class `smaAgri` resulting of executing the function `single_trial_analysis()`
`type`	A character string specifiying the type of plot. "summary", "correlation" or "spatial".
`filter_traits`	An optional character vector to filter traits.
`nudge_y_cv`	Vertical adjustment to nudge labels by when plotting CV bars. Only works if the argument type is "summary". 3 by default.

nudge_y_h2	Vertical adjustment to nudge labels by when plotting h2 bars. Only works if the argument type is "summary". 0.07 by default.
horizontal	If FALSE, the default, the labels are plotted vertically. If TRUE, the labels are plotted horizontally.
theme_size	Base font size, given in pts. 15 by default.
axis_size	Numeric input to define the axis size.
text_size	Numeric input to define the text size.
...	Further graphical parameters. For future improvements.

Value

A ggplot object.

Author(s)

Johan Aparicio [aut]

Examples

```
library(agridat)
library(agriutilities)
data(besag.met)
dat <- besag.met
results <- check_design_met(
  data = dat,
  genotype = "gen",
  trial = "county",
  traits = c("yield"),
  rep = "rep",
  block = "block",
  col = "col",
  row = "row"
)
out <- single_trial_analysis(results, progress = FALSE)
print(out)
plot(out, type = "summary", horizontal = TRUE)
plot(out, type = "correlation")
plot(out, type = "spatial")
```

---

print.checkAgri *Print an object of class* checkAgri

Description

Prints information about check_design_met() function.

Usage

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

Arguments

x	An object fitted with the function check_design_met().
...	Options used by the tibble package to format the output. See 'tibble::print()' for more details.

Value

an object inheriting from class checkAgri.

Author(s)

Johan Aparicio [aut]

Examples

```
library(agridat)
library(agriutilities)
data(besag.met)
dat <- besag.met
results <- check_design_met(
  data = dat,
  genotype = "gen",
  trial = "county",
  traits = c("yield"),
  rep = "rep",
  block = "block",
  col = "col",
  row = "row"
)
print(results)
```

---

print.metAgri *Print an object of class* metAgri

Description

Prints information about met_analysis() function.

Usage

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

Arguments

x	An object fitted with the function met_analysis().
...	Options used by the tibble package to format the output. See 'tibble::print()' for more details.

Value

an object inheriting from class metAgri.

Author(s)

Johan Aparicio [aut]

Examples

```
## Not run:
library(agridat)
library(agriutilities)
data(besag.met)
dat <- besag.met
results <- check_design_met(
  data = dat,
  genotype = "gen",
  trial = "county",
  traits = c("yield"),
  rep = "rep",
  block = "block",
  col = "col",
  row = "row"
)
out <- single_trial_analysis(results, progress = FALSE)
met_results <- met_analysis(out, progress = FALSE)
print(met_results)

## End(Not run)
```

---

print.smaAgri *Print an object of class* smaAgri

Description

Prints information about single_trial_analysis function.

Usage

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

Arguments

x	An object fitted with the function single_trial_analysis().
...	Options used by the tibble package to format the output. See 'tibble::print()' for more details.

Value

an object inheriting from class smaAgri.

Author(s)

Johan Aparicio [aut]

Examples

```
library(agridat)
library(agriutilities)
data(besag.met)
dat <- besag.met
results <- check_design_met(
  data = dat,
  genotype = "gen",
  trial = "county",
  traits = c("yield"),
  rep = "rep",
  block = "block",
  col = "col",
  row = "row"
)
out <- single_trial_analysis(results, progress = FALSE)
print(out)
```

---

single_trial_analysis *Single Trial Analysis*

Description

The results of the check_design_met() function are used in single_trial_analysis() to fit single trial models. This function can fit, Completely Randomized Designs (CRD), Randomized Complete Block Designs (RCBD), Resolvable Incomplete Block Designs (res-IBD), Non-Resolvable Row-Column Designs (Row-Col) and Resolvable Row-Column Designs (res-Row-Col).

Returns an object of class smaAgri, with a list of trial summary, BLUEs, BLUPs, heritability, variance components, potential extreme observations, residuals, the models fitted and the data used. This function will generate the required output to be used in the two-stage analysis.

Usage

```
single_trial_analysis(
  results = NULL,
  progress = TRUE,
  engine = "asreml",
  remove_outliers = FALSE
)
```

Arguments

results	Object of class checkAgri resulting of executing check_design_met() function.
progress	Should the progress of the modeling be printed. If TRUE, for every trial a line is output indicating the traits fitted for the particular trial.
engine	A character string specifying the name of the mixed modeling engine to use, either lme4 or asreml. For spatial designs, SpATS is always used, for other designs asreml as a default.
remove_outliers	Should outliers be removed? FALSE by default.

Value

An object of class smaAgri, with a list of:

fitted_models	A list containing the fitted models. (Both models, the one with Genotype as Random and the one with Genotype as Fixed)
resum_fitted_model	A data.frame containing a summary of the fitted models.
outliers	A data.frame containing extreme observations. If remove_outliers is TRUE, this data.frame will contain the observations removed.
blues_blups	A data.frame containing BLUPs/BLUEs for all the genotypes in each trial.
std_residuals	A data.frame containing the standardized residuals for the model with genotype as random component.
data	A data.frame containing the data used. If remove_outliers is TRUE, data will have missing values for the outliers detected.

Examples

```
library(agridat)
library(agriutilities)
data(besag.met)
dat <- besag.met
results <- check_design_met(
  data = dat,
  genotype = "gen",
  trial = "county",
  traits = c("yield"),
```

```
  rep = "rep",
  block = "block",
  col = "col",
  row = "row"
)
out <- single_trial_analysis(results, progress = FALSE)
print(out)
```

---

stability *Stability Coefficients*

Description

Stability Coefficients

Usage

```
stability(
  predictions = NULL,
  genotype = NULL,
  trial = NULL,
  response = NULL,
  best = "max"
)
```

Arguments

predictions	A data.frame with one value per GxE combination.
genotype	A character string indicating the column in predictions that contains genotypes.
trial	A character string indicating the column in predictions that contains trials.
response	A character string specifying the response variable.
best	A character string specifying how to define the best genotype by numeric value ("min", "max"). "max" by default.

Value

A data.frame with several stability measures. "superiority" (cultivar-superiority measure), "static" (Shukla's stability variance) and "wricke" (Wricke's ecovalence).

Examples

```
## Not run:
library(agridat)
library(agriutilities)

data(besag.met)
```

```
dat <- besag.met
results <- check_design_met(
  data = dat,
  genotype = "gen",
  trial = "county",
  traits = c("yield"),
  rep = "rep",
  block = "block",
  col = "col",
  row = "row"
)
out <- single_trial_analysis(results, progress = FALSE)
met_results <- met_analysis(out, progress = FALSE)

head(
  stability(
    predictions = met_results$BLUPs_GxE,
    genotype = "genotype",
    trial = "trial",
    response = "predicted.value"
  )
)

## End(Not run)
```

Index