Package 'metan'

January 14, 2020

Type Package

Title Multi Environment Trials Analysis

Version 1.2.1

Maintainer Tiago Olivoto <tiagoolivoto@gmail.com>

Description Performs stability analysis of multi-environment trial data using parametric and non-parametric methods. Parametric methods includes Additive Main Effects and Multiplicative Interaction (AMMI) analysis by Gauch (2013) <doi:10.2135/cropsci2013.04.0241>, Genotype plus Genotype-Environment (GGE) biplot analysis by Yan & Kang (2003) <doi:10.1201/9781420040371>, joint Regression Analysis by Eberhart & Russel (1966) (<doi:10.2135/cropsci1966.0011183X000600010011x>), ecovalence by Wricke (1965), genotypic confidence index by Annicchiarico (1992), Murakami & Cruz's (2004) method <doi:10.12702/1984-7033.v04n01a02>, stability variance by Shukla (1972) <doi:10.1038/hdy.1972.87>, weighted average of absolute scores by Olivoto et al. (2019a) <doi:10.2134/agronj2019.03.0220>, and multi-trait stability index by Olivoto et al. (2019b) <doi:10.2134/agronj2019.03.0221>. Non-parametric methods includes superiority index by Lin & Binns (1988) <doi:10.4141/cjps88-018>, nonparametric measures of phenotypic stability by Huehn (1990) <https://link.springer.com/article/10.1007/BF00024241>, TOP third statistic by Fox et al. (1990) <doi:10.1007/BF00040364>, geometric adaptability index described by Shahbazi (2019) <doi:10.1016/j.scienta.2019.04.047>. Functions for computing biometrical analysis such as path analysis, canonical correlation, partial correlation, clustering analysis, and tools for inspecting, manipulating, summarizing and plotting typical multi-environment trial data are also provided.

License GPL-3

URL https://github.com/TiagoOlivoto/metan

BugReports https://github.com/TiagoOlivoto/metan/issues

Depends R (>= 3.5.0)

Imports ade4, dendextend, cowplot, dplyr, FWDselect, GGally, ggforce, ggplot2, ggrepel, gplots, grid, lattice, lme4, lmerTest, magrittr, methods, progress, rlang, tibble, tidyr, tidyselect

Suggests DT, knitr, readxl, rmarkdown, roxygen2

VignetteBuilder knitr

Encoding UTF-8

Language en-US

LazyData true

RoxygenNote 7.0.2

NeedsCompilation no

Author Tiago Olivoto [aut, cre, cph] (<https://orcid.org/0000-0002-0241-9636>)

Repository CRAN

Date/Publication 2020-01-14 10:20:02 UTC

R topics documented:

`metan-package` *Multi-Environment Trial Analysis*

Description

metan provides functions for performing the most used analyses in the evaluation of multi-environment trials, including, but not limited to:

- ANOVA-based stability statistics;
- AMMI-based stability indexes;
- BLUP-based stability indexes;
- Cross-validation procedures for AMMI-family and BLUP models;
- GGE biplot analysis;
- Estimation using AMMI considering different numbers of interaction principal component axes;
- Graphics tools for generating biplots;
- Nonparametric stability statistics;
- Variance components and genetic parameters in mixed-effect models;
- Within-environment analysis of variance;

metan also provides functions for biometrical analysis such as path analysis, canonical correlation, partial correlation, clustering analysis, as well as tools for summarizing and plotting data.

A complete guide may be found at https://tiagoolivoto.github.io/metan/

`AMMI_indexes` *AMMI-based stability indexes*

Description

This function computes the following AMMI-based stability indexes: ASV, AMMI stability value (Purchase et al., 2000); SIPC, sums of the absolute value of the IPCA scores (Sneller et al. 1997); EV, averages of the squared eigenvector values (Sneller et al. 1997); and Za, absolute value of the relative contribution of IPCAs to the interaction (Zali et al. 2012), and WAAS, weighted average of absolute scores (Olivoto et al. 2019).

Usage

```
AMMI_indexes(.data, order.y = NULL, level = 0.95)
```

Arguments

`.data`	An object of class `waas` or `performs_ammi`
`order.y`	A vector of the same length of `x` used to order the response variable. Each element of the vector must be one of the `'h'` or `'l'`. If `'h'` is used, the response variable will be ordered from maximum to minimum. If `'l'` is used then the response variable will be ordered from minimum to maximum. Use a comma-separated vector of names. For example, `order.y = c("h,h,l,h,l")`.
`level`	The confidence level. Defaults to 0.95.

Details

The ASV index is computed as follows:

$$ASV_i = \left[\left[\frac{r\,\lambda_1^2}{r\,\lambda_2^2} \times (\lambda_1^{0.5}\,a_{i1}t_{j1})\right]^2 + (\lambda_2^{0.5}\,a_{i2}t_{j2})^2\right]^{0.5}$$

where r is the number of replications included in the analysis,

The SIPC index is computed as follows:

$$SIPC_i = \sum_{k=1}^{P} \left|\lambda_k^{0.5}\,a_{ik}\right|$$

where P is the number of IPCA retained via F-tests.

The EV index is computed as follows:

$$EV_i = \sum_{k=1}^{P} a_{ik}^2/P$$

The ZA index is computed as follows:

$$Za_i = \sum_{k=1}^{P} \theta_k a_{ik}$$

where θ_k is the percentage sum of squares explained by the kth IPCA.

$$WAAS_i = \sum_{k=1}^{p} |IPCA_{ik} \times EP_k| / \sum_{k=1}^{p} EP_k$$

where $WAAS_i$ is the weighted average of absolute scores of the *i*th genotype; PCA_{ik} is the score of the *i*th genotype in the *k*th IPCA; and EP_k is the explained variance of the *k*th IPCA for *k* = *1,2,..,p*, considering *p* the number of significant PCAs.

Five simultaneous selection indexes (ssi) are also computed by summation of the ranks of the ASV, SIPC, EV and Za indexes and the ranks of the mean yields (Farshadfar, 2008), which results in ssiASV, ssiSIPC, ssiEV, ssiZa, and ssiWAAS, respectively.

Value

A list where each element contains the result AMMI-based stability indexes for one variable.

Author(s)

Tiago Olivoto <tiagoolivoto@gmail.com>

References

Purchase, J.L., H. Hatting, and C.S. van Deventer. 2000. Genotype vs environment interaction of winter wheat (Triticum aestivum L.) in South Africa: II. Stability analysis of yield performance. South African J. Plant Soil 17:101-107. doi:10.1080/02571862.2000.10634878

Sneller, C.H., L. Kilgore-Norquest, and D. Dombek. 1997. Repeatability of Yield Stability Statistics in Soybean. Crop Sci. 37:383-390. doi:10.2135/cropsci1997.0011183X003700020013x

Zali, H., E. Farshadfar, S.H. Sabaghpour, and R. Karimizadeh. 2012. Evaluation of genotype vs environment interaction in chickpea using measures of stability from AMMI model. Ann. Biol. Res. 3:3126-3136. http://eprints.icrisat.ac.in/id/eprint/7173

Olivoto, T., A.D.C. L\'ucio, J.A.G. da silva, V.S. Marchioro, V.Q. de Souza, and E. Jost. 2019a. Mean performance and stability in multi-environment trials I: Combining features of AMMI and BLUP techniques. Agron. J. 111:2949-2960. doi:10.2134/agronj2019.03.0220

Examples

```
library(metan)
model <- waas(data_ge,
              env = ENV,
              gen = GEN,
              rep = REP,
              resp = c(GY, HM),
              verbose = FALSE)
model_indexes <- AMMI_indexes(model)


# Alternatively (and more intuitively) using %>%
```

```
res_ind <- data_ge %>%
           waas(ENV, GEN, REP, c(GY, HM)) %>%
           AMMI_indexes()
```

`Annicchiarico` *Annicchiarico's genotypic confidence index*

Description

Stability analysis using the known genotypic confidence index (Annicchiarico, 1992).

Usage

```
Annicchiarico(.data, env, gen, rep, resp, prob = 0.25, verbose = TRUE)
```

Arguments

`.data`	The dataset containing the columns related to Environments, Genotypes, replication/block and response variable(s)
`env`	The name of the column that contains the levels of the environments.
`gen`	The name of the column that contains the levels of the genotypes.
`rep`	The name of the column that contains the levels of the replications/blocks
`resp`	The response variable(s). To analyze multiple variables in a single procedure use, for example, `resp = c(var1,var2,var3)`.
`prob`	The probability of error assumed.
`verbose`	Logical argument. If `verbose = FALSE` the code will run silently.

Value

A list where each element is the result for one variable and contains the following data frames:

- **environments** Contains the mean, environmental index and classification as favorables and unfavorables environments.
- **general** Contains the genotypic confidence index considering all environments.
- **favorable** Contains the genotypic confidence index considering favorable environments.
- **unfavorable** Contains the genotypic confidence index considering unfavorable environments.

Author(s)

Tiago Olivoto, `<tiagoolivoto@gmail.com>`

References

Annicchiarico, P. 1992. Cultivar adaptation and recommendation from alfalfa trials in Northern Italy. J. Genet. Breed. 46:269-278.

See Also

superiority,ecovalence,ge_stats

Examples

```
library(metan)
Ann <- Annicchiarico(data_ge2,
                     env = ENV,
                     gen = GEN,
                     rep = REP,
                     resp = PH)
print(Ann)
```

---

anova_ind *Within-environment analysis of variance*

Description

This is a helper function that performs a within-environment analysis of variance and returns values such as Mean Squares, p-values, coefficient of variation, heritability, and accuracy of selection.

Usage

```
anova_ind(.data, env, gen, rep, resp, verbose = TRUE)
```

Arguments

.data	The dataset containing the columns related to Environments, Genotypes, replication/block and response variable(s).
env	The name of the column that contains the levels of the environments. The analysis of variance is computed for each level of this factor.
gen	The name of the column that contains the levels of the genotypes.
rep	The name of the column that contains the levels of the replications/blocks.
resp	The response variable(s). To analyze multiple variables in a single procedure a vector of variables may be used. For example resp = c(var1,var2,var3).
verbose	Logical argument. If verbose = FALSE the code will run silently.

Value

A list where each element is the result for one variable and contains:

- **individual** A data frame with the results of the individual analysis of variance.
- **MSRatio** The ratio between the higher and lower residual mean square.

Author(s)

Tiago Olivoto <tiagoolivoto@gmail.com>

Examples

```
library(metan)
# traditional usage approach
data = data_ge
anova1 = anova_ind(data_ge,
                   env = ENV,
                   gen = GEN,
                   rep = REP,
                   resp = GY)

# Using the pipe operator %>%
# Two variables, one run.
anova2 <- data_ge %>% anova_ind(ENV, GEN, REP, GY)
```

---

anova_joint *Joint analysis of variance*

Description

Performs a joint analysis of variance to check for the presence of genotype-vs-environment interactions.

Usage

```
anova_joint(.data, env, gen, rep, resp, verbose = TRUE)
```

Arguments

.data	The dataset containing the columns related to Environments, Genotypes, replication/block and response variable(s).
env	The name of the column that contains the levels of the environments. The analysis of variance is computed for each level of this factor.
gen	The name of the column that contains the levels of the genotypes.
rep	The name of the column that contains the levels of the replications/blocks.
resp	The response variable(s). To analyze multiple variables in a single procedure a vector of variables may be used. For example resp = c(var1,var2,var3).
verbose	Logical argument. If verbose = FALSE the code will run silently.

Value

A list where each element is the result for one variable containing the ANOVA table.

Author(s)

Tiago Olivoto <tiagoolivoto@gmail.com>

Examples

```
library(metan)
# traditional usage approach
anova1 = anova_joint(data_ge,
                     env = ENV,
                     gen = GEN,
                     rep = REP,
                     resp = GY)

# Using the pipe operator %>%
# Two variables, one run.
anova2 = data_ge %>% anova_joint(ENV, GEN, REP, c(GY, HM))
```

---

arrange_ggplot *Arrange multiple ggplot2 graphics in a single image window*

Description

This is a helper function to arrange ggplot2 objects in the metan package. It imports plot_grid. For a complete usability use that function.

Usage

```
arrange_ggplot(
  ...,
  plotlist = NULL,
  nrow = NULL,
  ncol = NULL,
  rel_widths = 1,
  rel_heights = 1,
  labels = NULL,
  hjust = -0.5,
  vjust = 1.5
)
```

Arguments

...	An object of class gg
plotlist	List of plots to display.
nrow, ncol	The number of rows and columns, respectively.

`rel_widths, rel_heights`	The Numerical vector of relative columns widths and rows heights, respectively.
`labels`	List of labels to be added to the plots.
`hjust, vjust`	Adjusts the horizontal and vertical position of each label.

Value

None.

Examples

```
library(ggplot2)
library(metan)
p1 <- ggplot(mtcars, aes(wt, mpg)) +
      geom_point()

p2 <- ggplot(mpg, aes(class, hwy)) +
             geom_boxplot()

arrange_ggplot(p1, p2)
```

---

as.lpcor *Coerce to an object of class lpcor*

Description

Functions to check if an object is of class `lpcor`, or coerce it if possible.

Usage

```
as.lpcor(...)
```

Arguments

`...`	A comma-separated list of matrices to be coerced to a list.

Value

An object of class `lpcor`.

Examples

```
library(metan)
library(dplyr)
mt_num = mtcars %>% select_if(., is.numeric)
lpdata = as.lpcor(cor(mt_num[1:5]),
                  cor(mt_num[1:5]),
```

```
                  cor(mt_num[2:6]),
                  cor(mt_num[4:8]))
is.lpcor(lpdata)
```

`as.split_factors` *Coerce to an object of class split_factors*

Description

Functions to coerce an object to a list of class `split_factors`, if possible.

Usage

```
as.split_factors(x, verbose = TRUE)
```

Arguments

`x`	The input data.
`verbose`	Logical argument. If `verbose = FALSE` the code will run silently.

Details

A dataframe may be easily coerced to be split into named subsets based on each combination of factors existing in the original dataframe. For example, if the original data has two columns, namely ENV (four levels) and HIB (ten levels), and ten numeric columns, then using `as.split_factors` will split the data into 40 10-columns subsets, corresponding to each combination of ENV x HIB.

Value

An object of class `split_factors`.

Examples

```
library(metan)
spdata = as.split_factors(iris)

spdata2 = as.split_factors(CO2)

is.split_factors(spdata2)
```

bind_cv *Bind cross-validation objects*

Description

Helper function that combines objects of class cv_ammi, cv_ammif or cv_blup. It is useful when looking for a boxplot containing the RMSPD values of those cross-validation procedures.

Usage

```
bind_cv(..., bind = "boot", sort = TRUE)
```

Arguments

...	Input objects of class cv_ammi, cv_ammif or cv_blup.
bind	What data should be used? To plot the RMSPD, use 'boot' (default). Use bind = 'means' to return the RMSPD mean for each model.
sort	Used to sort the RMSPD mean in ascending order.

Value

An object of class cv_ammif. The results will depend on the argument bind. If bind = 'boot' then the RMSPD of all models in ... will be bind to a unique data frame. If bind = 'means' then the RMSPD mean of all models in ... will be bind to an unique data frame.

Author(s)

Tiago Olivoto <tiagoolivoto@gmail.com>

Examples

```
library(metan)
# Two examples with only 5 resampling procedures
AMMI = cv_ammif(data_ge,
                resp = GY,
                gen = GEN,
                env = ENV,
                rep = REP,
                nboot = 5)
BLUP = cv_blup(data_ge,
               resp = GY,
               gen = GEN,
               env = ENV,
               rep = REP,
               nboot = 5)
bind_data = bind_cv(AMMI, BLUP)
plot(bind_data)
```

```
print(bind_cv(AMMI, BLUP, bind = 'means'))
```

---

can_corr *Canonical correlation analysis*

Description

Performs canonical correlation analysis with collinearity diagnostic, estimation of canonical loads, canonical scores, and hypothesis testing for correlation pairs.

Usage

```
can_corr(
  .data = NULL,
  FG = NULL,
  SG = NULL,
  by = NULL,
  means_by = NULL,
  use = "cor",
  test = "Bartlett",
  prob = 0.05,
  center = TRUE,
  stdscores = FALSE,
  verbose = TRUE,
  collinearity = TRUE
)
```

Arguments

.data	The data to be analyzed. Must be a dataframe containing the numeric variables that will be used in the estimation of the correlations. The data can also be passed directly by the arguments FG and SG. Alternatively, .data may be passed from the function split_factors. In such case, the canonical correlation will be estimated for each level of the grouping variable in that function.
FG, SG	If a dataframe is informed in .data, then FG and SG is a comma-separated list of unquoted variable names that will compose the first (smallest) and second (highest) group of the correlation analysis, respectively. Select helpers are also allowed.
by	One variable (factor) to split the data into subsets. The function is then applied to each subset and returns a list where each element contains the results for one level of the variable in by. To split the data by more than one factor variable, use the function split_factors to pass subsetted data to .data.

means_by	The argument means_by is a grouping variable to compute the means by. For example, if means_by = GEN, then the means of the numerical variables will be computed for each level of the grouping variable GEN, and the canonical correlation analysis will be computed using these means.
use	The matrix to be used. Must be one of 'cor' for analysis using the correlation matrix (default) or 'cov' for analysis using the covariance matrix.
test	The test of significance of the relationship between the FG and SG. Must be one of the 'Bartlett' (default) or 'Rao'.
prob	The probability of error assumed. Set to 0.05.
center	Should the data be centered to compute the scores?
stdscores	Rescale scores to produce scores of unit variance?
verbose	Logical argument. If TRUE (default) then the results are shown in the console.
collinearity	Logical argument. If TRUE (default) then a collinearity diagnostic is performed for each group of variables according to Olivoto et al.(2017).

Value

If .data is an object of class split_factors then the results will be returned into a list where each element has the following values.

- **Matrix** The correlation (or covariance) matrix of the variables
- **MFG, MSG** The correlation (or covariance) matrix for the variables of the first group or second group, respectively.
- **MFG_SG** The correlation (or covariance) matrix for the variables of the first group with the second group.
- **Coef_FG, Coef_SG** Matrix of the canonical coefficients of the first group or second group, respectively.
- Loads_FG, Loads_SG Matrix of the canonical loadings of the first group or second group, respectively.
- **Score_FG, Score_SG** Canonical scores for the variables in FG and SG, respectively.
- **Crossload_FG, Crossload_FG** Canonical cross-loadings for FG variables on the SG scores, and cross-loadings for SG variables on the FG scores, respectively.
- **SigTest** A dataframe with the correlation of the canonical pairs and hypothesis testing results.
- **collinearity** A list with the collinearity diagnostic for each group of variables.

Author(s)

Tiago Olivoto <tiagoolivoto@gmail.com>

References

Olivoto, T., V.Q. Souza, M. Nardino, I.R. Carvalho, M. Ferrari, A.J. Pelegrin, V.J. Szareski, and D. Schmidt. 2017. Multicollinearity in path analysis: a simple method to reduce its effects. Agron. J. 109:131-142. doi:10.2134/agronj2016.04.0196. 10.2134/agronj2016.04.0196

Examples

```
library(metan)

cc1 <- can_corr(data_ge2,
                FG = c(PH, EH, EP),
                SG = c(EL, ED, CL, CD, CW, KW, NR))

cc2 <- can_corr(FG = data_ge2[, 4:6],
                SG = data_ge2[, 7:13],
                verbose = FALSE,
                collinearity = FALSE)

# Canonical correlations for each environment
cc3 <- data_ge2 %>%
       can_corr(FG = c(PH, EH, EP),
                SG = c(EL, ED, CL, CD, CW, KW, NR),
                by = ENV,
                verbose = FALSE)
```

---

clustering *Clustering analysis*

Description

Performs clustering analysis with selection of variables.

Usage

```
clustering(
  .data,
  ...,
  by = NULL,
  means_by = NULL,
  scale = FALSE,
  selvar = FALSE,
  verbose = TRUE,
  distmethod = "euclidean",
  clustmethod = "average",
  nclust = NULL
)
```

Arguments

.data	The data to be analyzed. It may be a data frame containing the means of each observation in each variable or, alternatively, replicates for each factor. In this case,

	a grouping variable is required in the argument means_by to compute the means. In addition, .data may be an object passed from the function split_factors. In this case, the distances are computed for each level of the grouping variable.
...	The variables in .data to compute the distances. Set to NULL, i.e., all the numeric variables in .data are used.
by	One variable (factor) to split the data into subsets. The function is then applied to each subset and returns a list where each element contains the results for one level of the variable in by. To split the data by more than one factor variable, use the function split_factors to pass subsetted data to .data.
means_by	If .data doesn't contain the mean for each observation, then means_by is a grouping variable to compute the means. For example, if means_by = GEN, then the means of the numerical variables will be computed for each level of the grouping variable GEN.
scale	Should the data be scaled before computing the distances? Set to FALSE. If TRUE, then, each observation will be divided by the standard deviation of the variable Z_{ij} = X_{ij} / sd(j)
selvar	Logical argument, set to FALSE. If TRUE, then an algorithm for selecting variables is implemented. See the section **Details** for additional information.
verbose	Logical argument. If TRUE (default) then the results for variable selection are shown in the console.
distmethod	The distance measure to be used. This must be one of 'euclidean', 'maximum', 'manhattan', 'canberra', 'binary', 'minkowski', 'pearson', 'spearman', or 'kendall'. The last three are correlation-based distance.
clustmethod	The agglomeration method to be used. This should be one of 'ward.D', 'ward.D2', 'single', 'complete', 'average' (= UPGMA), 'mcquitty' (= WPGMA), 'median' (= WPGMC) or 'centroid' (= UPGMC).
nclust	The number of clusters to be formed. Set to NULL

Details

When selvar = TRUE a variable selection algorithm is executed. The objective is to select a group of variables that most contribute to explain the variability of the original data. The selection of the variables is based on eigenvalue/eigenvectors solution based on the following steps. **1:** compute the distance matrix and the co-optic correlation with the original variables (all numeric variables in dataset); **2:** compute the eigenvalues and eigenvectors of the correlation matrix between the variables; **3:** delete the variable with the largest weight (highest eigenvector in the lowest eigenvalue); **4:** compute the distance matrix and co-phenetic correlation with the remaining variables; **5:** compute the Mantel's correlation between the obtained distances matrix and the original distance matrix; **6:** iterate steps 2 to 5 p - 2 times, where p is the number of original variables. At the end of the p - 2 iterations, a summary of the models is returned. The distance is calculated with the variables that generated the model with the largest cophenetic correlation. I suggest a careful evaluation aiming at choosing a parsimonious model, i.e., the one with the fewer number of variables, that presents acceptable cophenetic correlation and high similarity with the original distances.

Value

- **data** The data that was used to compute the distances.

- **cutpoint** The cutpoint of the dendrogram according to Mojena (1977).
- **distance** The matrix with the distances.
- **de** The distances in an object of class `dist`.
- **hc** The hierarchical clustering.
- **Sqt** The total sum of squares.
- **tab** A table with the clusters and similarity.
- **clusters** The sum of square and the mean of the clusters for each variable.
- **cofgrap** If `selectvar = TRUE`, then, `cofpgrap` is a ggplot2-based graphic showing the cophenetic correlation for each model (with different number of variables). Else, will be a `NULL` object.
- **statistics** If `selectvar = TRUE`, then, `statistics` shows the summary of the models fitted with different number of variables, including cophenetic correlation, Mantel's correlation with the original distances (all variables) and the p-value associated with the Mantel's test. Else, will be a `NULL` object.

Author(s)

Tiago Olivoto <tiagoolivoto@gmail.com>

References

Mojena, R. 2015. Hierarchical grouping methods and stopping rules: an evaluation. Comput. J. 20:359-363. doi:10.1093/comjnl/20.4.359

Examples

```
library(metan)

# All rows and all numeric variables from data
d1 <- clustering(data_ge2)

# Based on the mean for each genotype
d2 <- clustering(data_ge2, means_by = GEN)

# Based on the mean of each genotype
# Variables NKR, TKW, and NKE
d3 <- clustering(data_ge2, NKR, TKW, NKE, means_by = GEN)

# Select variables for compute the distances
d4 <- clustering(data_ge2, means_by = GEN, selvar = TRUE)

# Compute the distances with standardized data
# Define 4 clusters
d5 <- clustering(data_ge2,
                 means_by = GEN,
                 scale = TRUE,
                 nclust = 4)
```

```
# Compute the distances for each environment
# Select the variables NKR, TKW, and NKE
# Use the mean for each genotype
d6 <- clustering(data_ge2,
                 NKR, TKW, NKE,
                 by = ENV,
                 means_by = GEN)

# Check the correlation between distance matrices
pairs_mantel(d6)
```

`colindiag` *Collinearity Diagnostics*

Description

Perform a (multi)collinearity diagnostic of a correlation matrix of predictor variables using several indicators, as shown by Olivoto et al. (2017).

Usage

```
colindiag(.data, ..., by = NULL, n = NULL, verbose = TRUE)
```

Arguments

`.data`	The data to be analyzed. Must be a symmetric correlation matrix or, a dataframe containing the predictor variables, or an object of class `split_factors`.
`...`	Variables to use in the correlation. If `...` is null then all the numeric variables from `.data` are used. It must be a single variable name or a comma-separated list of unquoted variables names.
`by`	One variable (factor) to split the data into subsets. The function is then applied to each subset and returns a list where each element contains the results for one level of the variable in `by`. To split the data by more than one factor variable, use the function `split_factors` to pass subsetted data to `.data`.
`n`	If a correlation matrix is provided, then `n` is the number of objects used to compute the correlation coefficients.
`verbose`	If `verbose = TRUE` then some results are shown in the console.

Value

The following values are returned. Please, note that if a grouping variable is used, then the results are returned into a list.

- **cormat** A symmetric Pearson's coefficient correlation matrix between the variables
- **corlist** A hypothesis testing for each of the correlation coefficients

- **evalevet** The eigenvalues with associated eigenvectors of the correlation matrix
- **VIF** The Variance Inflation Factors, being the diagonal elements of the inverse of the correlation matrix.
- **CN** The Condition Number of the correlation matrix, given by the ratio between the largest and smallest eigenvalue.
- **det** The determinant of the correlation matrix.
- **largest_corr** The largest correlation (in absolute value) observed.
- **smallest_corr** The smallest correlation (in absolute value) observed.
- **weight_var** The variables with largest eigenvector (largest weight) in the eigenvalue of smallest value, sorted in decreasing order.

Author(s)

Tiago Olivoto <tiagoolivoto@gmail.com>

References

Olivoto, T., V.Q. Souza, M. Nardino, I.R. Carvalho, M. Ferrari, A.J. Pelegrin, V.J. Szareski, and D. Schmidt. 2017. Multicollinearity in path analysis: a simple method to reduce its effects. Agron. J. 109:131-142. doi:10.2134/agronj2016.04.0196. doi:10.2134/agronj2016.04.0196

Olivoto, T., M. Nardino, I.I.R. Carvalho, D.N. Follmann, M. Ferrari, A.J. de Pelegrin, V.J. Szareski, A.C. de Oliveira, B.O. Caron, and V.Q. de Souza. 2017. Optimal sample size and data arrangement method in estimating correlation matrices with lesser collinearity: A statistical focus in maize breeding. African J. Agric. Res. 12:93-103. doi:10.5897/AJAR2016.11799.

Examples

```
# Using the correlation matrix
library(metan)

cor_iris <- cor(iris[,1:4])
n <- nrow(iris)

col_diag <- colindiag(cor_iris, n = n)


# Using a data frame
col_diag_gen <- data_ge2 %>%
                split_factors(GEN) %>%
                colindiag()

# Diagnostic by levels of a factor selecting desired variables
col_diag_gen <- data_ge2 %>%
                split_factors(GEN) %>%
                colindiag(EH, PH, CD, CL)
```

comb_vars *Pairwise combinations of variables*

Description

Pairwise combinations of variables that will be the result of a function applied to each combination.

Usage

```
comb_vars(.data, order = "first", FUN = "+", verbose = TRUE)
```

Arguments

.data	A matrix of data with, say, p columns.
order	The order on how the results will appear in the output. Default is order = 'first'. In this case, assuming that .data has four columns, namely, V1, V2, V3, V4, the order of columns in the output will be V1.V2, V1.V3, V1.V4, V2.V3, V2.V4, V3.V4. If order = 'second', the result will be then V1.V2, V1.V3, V2.V3, V1.V4, V2.V4, V3.V4.
FUN	The function that will be applied to each combination. The default is +, i.e., V1 + V2.
verbose	Logical argument. If verbose = FALSE the code will run silently.

Value

A data frame containing all possible combination of variables. Each combination is the result of the function in FUN applied to the two variables.

Author(s)

Tiago Olivoto <tiagoolivoto@gmail.com>

Examples

```
library(metan)
data <- data.frame(A = rnorm(n = 5, mean = 10, sd = 3),
                   B = rnorm(n = 5, mean = 120, sd = 30),
                   C = rnorm(n = 5, mean = 40, sd = 10),
                   D = rnorm(n = 5, mean = 1100, sd = 200),
                   E = rnorm(n = 5, mean = 2, sd = 1))
comb1 <- comb_vars(data)
comb2 <- comb_vars(data, FUN = '*', order = 'second')
```

corr_ci *Confidence interval for correlation coefficient*

Description

Computes the half-width confidence interval for correlation coefficient using the nonparametric method proposed by Olivoto et al. (2018).

Usage

```
corr_ci(.data = NA, ..., r = NULL, n = NULL, by = NULL, verbose = TRUE)
```

Arguments

.data	A dataset containing variables only or a symmetric correlation matrix.
...	Variables to compute the confidence interval. If not informed, all the numeric variables from .data are used.
r	If data is not available, provide the value for correlation coefficient.
n	The sample size if data is a correlation matrix or if r is informed.
by	One variable (factor) to split the data into subsets. The function is then applied to each subset and returns a list where each element contains the results for one level of the variable in by. To split the data by more than one factor variable, use the function split_factors to pass subsetted data to .data.
verbose	If verbose = TRUE then some results are shown in the console.

Details

The half-width confidence interval is computed according to the following equation:

$$CI_w = 0.45304^r \times 2.25152 \times n^{-0.50089}$$

where n is the sample size and r is the correlation coefficient.

Value

A tibble containing the values of the correlation, confidence interval, upper and lower limits for all combination of variables.

Author(s)

Tiago Olivoto <tiagoolivoto@gmail.com>

References

Olivoto, T., A.D.C. Lucio, V.Q. Souza, M. Nardino, M.I. Diel, B.G. Sari, D.. K. Krysczun, D. Meira, and C. Meier. 2018. Confidence interval width for Pearson's correlation coefficient: a Gaussian-independent estimator based on sample size and strength of association. Agron. J. 110:1-8. 10.2134/agronj2016.04.0196

Examples

```
library(metan)

CI1 <- corr_ci(data_ge2)

# By each level of the factor 'ENV'
CI2 <- corr_ci(data_ge2, CD, TKW, NKE, by = ENV)
```

---

corr_coef *Computes Pearson's correlation matrix with p-values*

Description

Computes Pearson's correlation matrix with p-values

Usage

```
corr_coef(data, ..., verbose = TRUE)
```

Arguments

data	The data set.
...	Variables to use in the correlation. If no variable is informed all the numeric variables from data are used.
verbose	Logical argument. If verbose = FALSE the code is run silently.

Value

A list with the correlation coefficients and p-values

Author(s)

Tiago Olivoto <tiagoolivoto@gmail.com>

Examples

```
library(metan)

# All numeric variables
all <- corr_coef(data_ge2)

# Select variables
sel <- corr_coef(data_ge2, EP, EL, CD, CL)
print(sel)
```

corr_plot *Visualization of a correlation matrix*

Description

Graphical and numerical visualization of a correlation matrix

Usage

```
corr_plot(
  .data,
  ...,
  upper = "corr",
  lower = "scatter",
  axis.labels = FALSE,
  show.labels.in = "show",
  size.axis.label = 12,
  diag = TRUE,
  diag.type = "histogram",
  bins = 20,
  col.diag = "gray",
  alpha.diag = 1,
  col.up.panel = "gray",
  col.lw.panel = "gray",
  col.dia.panel = "gray",
  prob = 0.05,
  col.sign = "green",
  alpha.sign = 0.15,
  lab.position = "tr",
  progress = NULL,
  smooth = FALSE,
  col.smooth = "red",
  size.smooth = 0.3,
  confint = TRUE,
  size.point = 1,
  shape.point = 19,
  alpha.point = 0.7,
  fill.point = NULL,
  col.point = "black",
  minsize = 2,
  maxsize = 3,
  pan.spacing = 0.15,
  digits = 2,
  export = FALSE,
  file.type = "pdf",
  file.name = NULL,
  width = 8,
```

```
  height = 7,
  resolution = 300
)
```

Arguments

`.data`	The data. Should, preferentially, contain numeric variables only. If `.data` has factor-columns, these columns will be deleted with a warning message.
`...`	Variables to use in the correlation. If no variable is informed all the numeric variables from `.data` are used.
`upper`	The visualization method for the upper triangular correlation matrix. Must be one of `'corr'` (numeric values), `'scatter'` (the scatterplot for each pairwise combination), or `NULL` to set a blank diagonal.
`lower`	The visualization method for the lower triangular correlation matrix. Must be one of `'corr'` (numeric values), `'scatter'` (the scatterplot for each pairwise combination), or `NULL` to set a blank diagonal.
`axis.labels`	Should the axis labels be shown in the plot? Set to `FALSE`.
`show.labels.in`	Where to show the axis labels. Defaults to "show" bottom and left. Use "diag" to show the labels on the diagonal. In this case, the diagonal layer (boxplot, density or histogram) will be overwritten.
`size.axis.label`	The size of the text for axis labels if `axis.labels = TRUE`. Defaults to 12.
`diag`	Should the diagonal be shown?
`diag.type`	The type of plot to show in the diagonal if `diag TRUE`. It must be one of the 'histogram' (to show an histogram), 'density' to show the Kernel density, or 'boxplot' (to show a boxplot).
`bins`	The number of bins, Defaults to 20.
`col.diag`	If `diag = TRUE` then `diagcol` is the color for the distribution. Set to gray.
`alpha.diag`	Alpha-transparency scale [0-1] to make the diagonal plot transparent. 0 = fully transparent; 1 = full color. Set to 0.15
`col.up.panel, col.lw.panel, col.dia.panel`	The color for the upper, lower, and diagonal panels, respectively. Set to 'gray'.
`prob`	The probability of error. Significant correlations will be highlighted with '*', '**', and '***' (0.05, 0.01, and 0.001, respectively). Scatterplots with significant correlations may be color-highlighted.
`col.sign`	The color that will highlight the significant correlations. Set to 'green'.
`alpha.sign`	Alpha-transparency scale [0-1] to make the plot area transparent. 0 = fully transparent; 1 = full color. Set to 0.15
`lab.position`	The position that the labels will appear. Set to `'tr'`, i.e., the legends will appear in the top and right of the plot. Other allowed options are `'tl'` (top and left), `'br'` (bottom and right), `'bl'` (bottom and left).
`progress`	`NULL` (default) for a progress bar in interactive sessions with more than 15 plots, `TRUE` for a progress bar, `FALSE` for no progress bar.
`smooth`	Should a linear smooth line be shown in the scatterplots? Set to `FALSE`.

col.smooth	The color for the smooth line.
size.smooth	The size for the smooth line.
confint	Should a confidence band be shown with the smooth line? Set to TRUE.
size.point	The size of the points in the plot. Set to 0.5.
shape.point	The shape of the point, set to 1.
alpha.point	Alpha-transparency scale [0-1] to make the points transparent. 0 = fully transparent; 1 = full color. Set to 0.7
fill.point	The color to fill the points. Valid argument if points are between 21 and 25.
col.point	The color for the edge of the point, set to black.
minsize	The size of the letter that will represent the smallest correlation coefficient.
maxsize	The size of the letter that will represent the largest correlation coefficient.
pan.spacing	The space between the panels. Set to 0.15.
digits	The number of digits to show in the plot.
export	Logical argument. If TRUE, then the plot is exported to the current directory.
file.type	The format of the file if export = TRUE. Set to 'pdf'. Other possible values are *.tiff using file.type = 'tiff'.
file.name	The name of the plot when exported. Set to NULL, i.e., automatically.
width	The width of the plot, set to 8.
height	The height of the plot, set to 7.
resolution	The resolution of the plot if file.type = 'tiff' is used. Set to 300 (300 dpi).

Value

An object of class gg,ggmatrix.

Author(s)

Tiago Olivoto <tiagoolivoto@gmail.com>

Examples

```
library(metan)
dataset = data_ge2

# Default plot setting
corr_plot(dataset)

# Chosing variables to be correlated
corr_plot(dataset, CD, EL, PERK, NKR)

# Changing the layout
corr_plot(dataset, CD, EL, PERK, NKR,
          lower = NULL,
          upper = 'corr')
```

```
# Axis labels, similar to the function pairs()
# Gray scale
corr_plot(dataset, CD, EL, PERK, NKR,
          shape.point = 19,
          size.point = 2,
          alpha.point = 0.5,
          alpha.diag = 0,
          pan.spacing = 0,
          col.sign = 'gray',
          alpha.sign = 0.3,
          axis.labels = TRUE)

corr_plot(dataset, CD, EL, PERK, NKR, CW, NKE,
          prob = 0.01,
          shape.point = 21,
          col.point = 'black',
          fill.point = 'orange',
          size.point = 2,
          alpha.point = 0.6,
          maxsize = 4,
          minsize = 2,
          smooth = TRUE,
          size.smooth = 1,
          col.smooth = 'black',
          col.sign = 'cyan',
          col.up.panel = 'black',
          col.lw.panel = 'black',
          col.dia.panel = 'black',
          pan.spacing = 0,
          lab.position = 'tl')
```

`corr_ss` *Sample size planning for a desired Pearson's correlation confidence interval*

Description

Find the required (sufficient) sample size for computing a Pearson correlation coefficient with a desired confidence interval (Olivoto et al., 2018).

Usage

```
corr_ss(r, CI, verbose = TRUE)
```

Arguments

`r`	The magnitude of the correlation coefficient.
`CI`	The half-width for confidence interval at $p < 0.05$.
`verbose`	Logical argument. If `verbose = FALSE` the code is run silently.

Details

The required (sufficient) sample size is computed as follows:

$$n = [CI_w/0.45304^r \times 2.25152]^{-0.50089}$$

where CI_w is desired confidence interval and `r` is the correlation coefficient.

Author(s)

Tiago Olivoto <tiagoolivoto@gmail.com>

References

Olivoto, T., A.D.C. Lucio, V.Q. Souza, M. Nardino, M.I. Diel, B.G. Sari, D.. K. Krysczun, D. Meira, and C. Meier. 2018. Confidence interval width for Pearson's correlation coefficient: a Gaussian-independent estimator based on sample size and strength of association. Agron. J. 110:1-8. 10.2134/agronj2016.04.0196

Examples

```
corr_ss(r = 0.60, CI = 0.1)
```

---

corr_stab_ind *Correlation between stability indexes*

Description

Computes the Spearman's rank correlation between the parametric and nonparametric stability indexes computed with the function `ge_stats`.

Usage

```
corr_stab_ind(x, stats = "all", plot = TRUE, ...)
```

Arguments

`x`	An object of class `ge_stats`.
`stats`	The statistics to compute the correlation. See the section **Details** for more information.
`plot`	Plot the heat map with the correlations? Defaults to `TRUE`.
`...`	Other arguments to be passed to the function `plot.corr_coef`.

Details

The argument `stats` is used to chose the statistics to show the ranks. Allowed values are `"all"` (All statistics, default), `"par"` (Parametric statistics), `"nonpar"` (Non-parametric statistics), `"ammi"` (AMMI-based stability statistics), or the following values that can be combined into comma-separated character vector. `"Y"` (Response variable), `"Var"` (Genotype's variance), `"Shukla"` (Shukla's variance), `"Wi_g"`, `"Wi_f"`, `"Wi_u"` (Annichiarrico's genotypic confidence index for all, favorable and unfavorable environments, respectively), `"Ecoval"` (Wricke's ecovalence), `"Sij"` (Deviations from the joint-regression analysis), `"R2"` (R-squared from the joint-regression analysis), `"ASV"` (AMMI-stability value), `"SIPC"` (sum of the absolute values of the IPCA scores), `"EV"` (Average of the squared eigenvector values), `"ZA"` (Absolute values of the relative contributions of the IPCAs to the interaction), `"WAAS"` (Weighted Average of Absolute Scores), `"HMGV"` (Harmonic mean of the genotypic value), `"RPGV"` (Relative performance of the genotypic values), `"HMRPGV"` (Harmonic mean of the relative performance of the genotypic values), `"Pi_a"`, `"Pi_f"`, `"Pi_u"` (Superiority indexes for all, favorable and unfavorable environments, respectively), `"Gai"` (Geometric adaptability index), `"S1"` (mean of the absolute rank differences of a genotype over the n environments), `"S2"` (variance among the ranks over the k environments), `"S3"` (sum of the absolute deviations), `"S6"` (relative sum of squares of rank for each genotype), `"N1"`, `"N2"`, `"N3"`, `"N4"` (Thennarasu"s statistics)).

Value

A list with the data (ranks) correlation, p-values and a heat map showing the correlation coefficients.

Author(s)

Tiago Olivoto <tiagoolivoto@gmail.com>

Examples

```
library(metan)
model <- ge_stats(data_ge, ENV, GEN, REP, GY)
a <- corr_stab_ind(model)
b <- corr_stab_ind(model, stats = "ammi")
c <- corr_stab_ind(model, stats = c("ASV, Sij, R2, WAAS, N1"))
```

---

covcor_design *Variance-covariance matrices for designed experiments*

Description

Compute variance-covariance and correlation matrices using data from a designed (RCBD or CRD) experiment.

Usage

```
covcor_design(.data, gen, rep, resp, design = "RCBD", by = NULL, type = NULL)
```

Arguments

.data	The dataset containing the columns related to Genotypes, replication/block and response variables. Alternatively, it is possible to use an object of class 'split_factors' to compute the results for each level of the grouping factor. See ?split_factors.
gen	The name of the column that contains the levels of the genotypes.
rep	The name of the column that contains the levels of the replications/blocks.
resp	The response variables. For example resp = c(var1,var2,var3).
design	The experimental design. Must be RCBD or CRD.
by	One variable (factor) to split the data into subsets. The function is then applied to each subset and returns a list where each element contains the results for one level of the variable in by. To split the data by more than one factor variable, use the function split_factors to pass subsetted data to .data.
type	What the matrices should return? Set to NULL, i.e., a list of matrices is returned. The argument type allow the following values 'pcor','gcor','rcor', (which will return the phenotypic, genotypic and residual correlation matrices, respectively) or 'pcov','gcov','rcov' (which will return the phenotypic, genotypic and residual variance-covariance matrices, respectively). Alternatively, it is possible to get a matrix with the means of each genotype in each trait, by using type = 'means'.

Value

An object of class covcor_design containing the following items:

- **geno_cov** The genotypic covariance.
- **phen_cov** The phenotypic covariance.
- **resi_cov** The residual covariance.
- **geno_cor** The phenotypic correlation.
- **phen_cor** The phenotypic correlation.
- **resi_cor** The residual correlation.

If .data is an object of class split_factors then the output will be a list with the above values for each grouping variable in the function split_factors to pass subsetted data.to pass subsetted data to code.data.to pass subsetted data to code.data.

Author(s)

Tiago Olivoto <tiagoolivoto@gmail.com>

Examples

```
library(metan)
# List of matrices
data <- subset(data_ge2, ENV == 'A1')
matrices <- covcor_design(data, gen = GEN, rep = REP,
```

```
                          resp = c(PH, EH, NKE, TKW))

# Genetic correlations
gcor <- covcor_design(data,
                      gen = GEN,
                      rep = REP,
                      resp = c(PH, EH, NKE, TKW),
                      type = 'gcor')

# Residual (co)variance matrix for each environment
rcov <- covcor_design(data_ge2,
                      gen = GEN,
                      by = ENV,
                      rep = REP,
                      resp = c(PH, EH, CD, CL),
                      type = "rcov")
```

`cv_ammi` *Cross-validation procedure*

Description

Cross-validation for estimation of AMMI models

Usage

```
cv_ammi(
  .data,
  env,
  gen,
  rep,
  resp,
  block = NULL,
  naxis = 2,
  nboot = 200,
  design = "RCBD",
  verbose = TRUE
)
```

Arguments

`.data`	The dataset containing the columns related to Environments, Genotypes, replication/block and response variable(s).
`env`	The name of the column that contains the levels of the environments.
`gen`	The name of the column that contains the levels of the genotypes.
`rep`	The name of the column that contains the levels of the replications/blocks.

resp	The response variable.
block	Defaults to NULL. In this case, a randomized complete block design is considered. If block is informed, then a resolvable alpha-lattice design (Patterson and Williams, 1976) is employed. **All effects are assumed to be fixed.**
naxis	The number of axis to be considered for estimation of GE effects.
nboot	The number of resamples to be used in the cross-validation. Defaults to 200.
design	The experimental design. Defaults to RCBD (Randomized complete Block Design). For Completely Randomized Designs inform design = 'CRD'.
verbose	A logical argument to define if a progress bar is shown. Default is TRUE.

Details

THe original dataset is split into two datasets: training set and validation set. The 'training' set has all combinations (genotype x environment) with N-1 replications. The 'validation' set has the remaining replication. The splitting of the dataset into modeling and validation sets depends on the design informed. For Completely Randomized Block Design (default), and alpha-lattice design (declaring block arguments), complete replicates are selected within environments. The remained replicate serves as validation data. If design = 'RCD' is informed, completely randomly samples are made for each genotype-by-environment combination (Olivoto et al. 2019). The estimated values considering naxis-Interaction Principal Component Axis are compared with the 'validation' data. The Root Mean Square Prediction Difference (RMSPD) is computed. At the end of boots, a list is returned.

Value

An object of class cv_ammi with the following items: * **RMSPD**: A vector with nboot-estimates of the Root Mean Squared Prediction Difference between predicted and validating data.

- **RMSPDmean**: The mean of RMSPDmean estimates.
- **Estimated**: A data frame that contain the values (predicted, observed, validation) of the last loop.
- **Modeling**: The dataset used as modeling data in the last loop
- **Testing**: The dataset used as testing data in the last loop.

Author(s)

Tiago Olivoto <tiagoolivoto@gmail.com>

References

Olivoto, T., A.D.C. L\'ucio, J.A.G. da silva, V.S. Marchioro, V.Q. de Souza, and E. Jost. 2019. Mean performance and stability in multi-environment trials I: Combining features of AMMI and BLUP techniques. Agron. J. 111:2949-2960. doi:10.2134/agronj2019.03.0220

Patterson, H.D., and E.R. Williams. 1976. A new class of resolvable incomplete block designs. Biometrika 63:83-92.

See Also

cv_ammif,cv_blup

Examples

```
library(metan)
model <- cv_ammi(data_ge,
                 env = ENV,
                 gen = GEN,
                 rep = REP,
                 resp = GY,
                 nboot = 10,
                 naxis = 2)

# Alternatively using the pipe operator %>%
model <- data_ge %>%
         cv_ammi(ENV, GEN, REP, GY)
```

---

cv_ammif *Cross-validation procedure*

Description

Cross-validation for estimation of all AMMI-family models

Usage

```
cv_ammif(
  .data,
  env,
  gen,
  rep,
  resp,
  nboot = 200,
  block,
  design = "RCBD",
  verbose = TRUE
)
```

Arguments

.data	The dataset containing the columns related to Environments, Genotypes, replication/block and response variable(s).

env	The name of the column that contains the levels of the environments.
gen	The name of the column that contains the levels of the genotypes.
rep	The name of the column that contains the levels of the replications/blocks.
resp	The response variable.
nboot	The number of resamples to be used in the cross-validation. Defaults to 200.
block	Defaults to NULL. In this case, a randomized complete block design is considered. If block is informed, then a resolvable alpha-lattice design (Patterson and Williams, 1976) is employed. **All effects are assumed to be fixed.**
design	The experimental design used in each environment. Defaults to RCBD (Randomized complete Block Design). For Completely Randomized Designs inform design = 'CRD'.
verbose	A logical argument to define if a progress bar is shown. Default is TRUE.

Details

cv_ammif provides a complete cross-validation of replicate-based data using AMMI-family models. By default, the first validation is carried out considering the AMMIF (all possible axis used). Considering this model, the original dataset is split up into two datasets: training set and validation set. The 'training' set has all combinations (genotype x environment) with N-1 replications. The 'validation' set has the remaining replication. The splitting of the dataset into modeling and validation sets depends on the design informed. For Completely Randomized Block Design (default), and alpha-lattice design (declaring block arguments), complete replicates are selected within environments. The remained replicate serves as validation data. If design = 'RCD' is informed, completely randomly samples are made for each genotype-by-environment combination (Olivoto et al. 2019). The estimated values for each member of the AMMI-family model are compared with the 'validation' data. The Root Mean Square Prediction Difference (RMSPD) is computed. At the end of boots, a list is returned.

Value

An object of class cv_ammif with the following items:

- **RMSPD**: A vector with nboot-estimates of the Root Mean Squared Prediction Difference between predicted and validating data.
- **RMSPDmean**: The mean of RMSPDmean estimates.
- **Estimated**: A data frame that contain the values (predicted, observed, validation) of the last loop.
- **Modeling**: The dataset used as modeling data in the last loop
- **Testing**: The dataset used as testing data in the last loop.

Author(s)

Tiago Olivoto <tiagoolivoto@gmail.com>

References

Patterson, H.D., and E.R. Williams. 1976. A new class of resolvable incomplete block designs. Biometrika 63:83-92.

See Also

cv_ammi,cv_blup

Examples

```
library(metan)
model <- cv_ammif(data_ge,
                  env = ENV,
                  gen = GEN,
                  rep = REP,
                  resp = GY,
                  nboot = 10)

# Alternatively (and more intuitively) using the pipe operator %>%
model <- data_ge %>%
         cv_ammif(ENV, GEN, REP, GY, 10)
```

---

cv_blup *Cross-validation procedure*

Description

Cross-validation for blup prediction.

Usage

```
cv_blup(
  .data,
  env,
  gen,
  rep,
  resp,
  block = NULL,
  nboot = 200,
  random = "gen",
  verbose = TRUE
)
```

Arguments

.data	The dataset containing the columns related to Environments, Genotypes, replication/block and response variable(s).
env	The name of the column that contains the levels of the environments.
gen	The name of the column that contains the levels of the genotypes.

rep	The name of the column that contains the levels of the replications/blocks.
resp	The response variable.
block	Defaults to NULL. In this case, a randomized complete block design is considered. If block is informed, then a resolvable alpha-lattice design (Patterson and Williams, 1976) is employed. See how fixed and random effects are considered, see the section **Details**.
nboot	The number of resamples to be used in the cross-validation. Defaults to 200
random	The effects of the model assumed to be random. See **Details** for more information.
verbose	A logical argument to define if a progress bar is shown. Default is TRUE.

Details

This function provides a cross-validation procedure for mixed models using replicate-based data. By default, complete blocks are randomly selected within each environment. In each iteration, the original dataset is split up into two datasets: training and validation data. The 'training' set has all combinations (genotype x environment) with R - 1 replications. The 'validation' set has the remaining replication. The estimated values are compared with the 'validation' data and the Root Means Square Prediction Difference (Olivoto et al. 2019) is computed. At the end of boots, a list is returned.

Six models may be fitted depending upon the values in `block` and `random` arguments. * **Model 1:** `block = NULL` and `random = "gen"` (The default option). This model considers a Randomized Complete Block Design assuming genotype and genotype-vs-environment as random effects. Environment and blocks nested within environments are treated as fixed factors.

- **Model 2:** `block = NULL` and `random = "env"`. This model considers a Randomized Complete Block Design treating environment, genotype-vs-environment, and blocks-within-environments as random factors. Genotypes are assumed to be fixed factors.
- **Model 3:** `block = NULL` and `random = "all"`. This model considers a Randomized Complete Block Design assuming all effects (genotypes, environments, genotype-vs-environment interaction and blocks nested within environments) as random.
- **Model 4:** `block != NULL` and `random = "gen"`. This model considers an alpha-lattice design assuming genotype, genotype-vs-environment interaction, and incomplete block nested within replicates as random to make use of inter-block information (Mohring et al., 2015). Complete replicates nested within environments and environments are treated as fixed factors.
- **Model 5:** `block != NULL` and `random = "env"`. This model considers an alpha-lattice design assuming genotype as fixed. All other sources of variation (environment, complete replicates nested within environments, and incomplete blocks nested within replicates) as treated as random factors.
- **Model 6:** `block != NULL` and `random = "all"`. This model considers an alpha-lattice design assuming all effects, except the intercept, as random factors.

Value

An object of class `cv_blup` with the following items: * **RMSPD**: A vector with nboot-estimates of the root mean squared prediction difference between predicted and validating data. * **RMSPDmean** The mean of RMSPDmean estimates.

Author(s)

Tiago Olivoto <tiagoolivoto@gmail.com>

References

Olivoto, T., A.D.C. L\'ucio, J.A.G. da silva, V.S. Marchioro, V.Q. de Souza, and E. Jost. 2019. Mean performance and stability in multi-environment trials I: Combining features of AMMI and BLUP techniques. Agron. J. 111:2949-2960. doi:10.2134/agronj2019.03.0220

Patterson, H.D., and E.R. Williams. 1976. A new class of resolvable incomplete block designs. Biometrika 63:83-92.

Mohring, J., E. Williams, and H.-P. Piepho. 2015. Inter-block information: to recover or not to recover it? TAG. Theor. Appl. Genet. 128:1541-54. doi:10.1007/s00122-015-2530-0

See Also

cv_ammi, cv_ammif

Examples

```
library(metan)
model <- cv_blup(data_ge,
                 env = ENV,
                 gen = GEN,
                 rep = REP,
                 resp = GY,
                 nboot = 10)

# Alternatively using the pipe operator %>%
model <- data_ge %>%
         cv_blup(ENV, GEN, REP, GY, nboot = 10)
```

---

data_alpha *Data from an alpha lattice design*

Description

Alpha lattice design of spring oats

Format

A tibble with 72 observations on the following 5 variables.

- **PLOT** Plot number
- **REP** Replicate code

- **BLOCK** Incomplete block code
- **GEN** Genotype code
- **YIELD** Observed dry matter yield (tonnes/ha)

Details

A spring oats trial grown in Craibstone. There were 24 varieties in 3 replicates, each consisting of 6 incomplete blocks of 4 plots. Planted in a resolvable alpha design. The plots were laid out in a single line.

Author(s)

Tiago Olivoto <tiagoolivoto@gmail.com>

Source

J. A. John & E. R. Williams (1995). Cyclic and computer generated designs, Chapman and Hall, London. Page 146.

---

data_g *Single maize trial*

Description

This dataset contain data on 15 traits assessed in 13 maize hybrids. The experimental design was a RCBD with 3 blocks and 1 replications per block. It is used as an example in the function `gamem` of the **metan** package.

Format

A tibble with 39 observations on the following 17 variables.

- **GEN** A factor with 13 levels; each level represents one maize hybrid.
- **REP** A factor with 3 levels; each level represents one replication/block.
- **PH** Plant height, in cm.
- **EH** Ear height, in cm.
- **EP** Ear position, i.e., the ratio EH/PH.
- **EL** Ear length, in cm.
- **ED** Ear diameter, in mm.
- **CL** Cob length, in cm.
- **CD** Cob diameter, in mm.
- **CW** Cob weight, in g.
- **KW** Kernel weight, in cm.
- **NR** Number of rows.

- **NKR** Number of kernels per row.
- **CDED** Cob diameter / Ear diameter ratio.
- **PERK** Percentage of kernels.
- **TKW** Thousand-kernel weight
- **NKE** Number of kernels per row.

Author(s)

Tiago Olivoto <tiagoolivoto@gmail.com>

Source

Personal data

`data_ge` *Multi-environment trial of oat*

Description

This dataset contain data on two variables assessed in 10 genotypes growing in in 11 environments. The experimental design was a RCBD with 3 replicates(blocks). This data provide examples for several functions of **metan** package.

Format

A tibble with 420 observations on the following 5 variables.

- **ENV** A factor with 14 levels; each level represents one cultivation environment.
- **GEN** A factor with 10 levels; each level represents one genotype.
- **REP** A factor with 3 levels; each level represents one replication/block.
- **GY** A continuous variable (grain yield) observed in each plot.
- **HM** A continuous variable (hectoliter mass) observed in each plot.

Author(s)

Tiago Olivoto <tiagoolivoto@gmail.com>

Source

Personal data

`data_ge2` *Multi-environment trial of maize*

Description

This dataset contain data on 15 traits assessed in 13 maize hybrids growing in 4 environments. The experimental design was a RCBD with 3 blocks and 1 replications per block. It may be used as example in several functions of **metan** package.

Format

A tibble with 156 observations on the following 18 variables.

- **ENV** A factor with 4 levels; each level represents one cultivation environment.
- **GEN** A factor with 13 levels; each level represents one maize hybrid.
- **REP** A factor with 3 levels; each level represents one replication/block.
- **PH** Plant height, in cm.
- **EH** Ear height, in cm.
- **EP** Ear position, i.e., the ratio EH/PH.
- **EL** Ear length, in cm.
- **ED** Ear diameter, in mm.
- **CL** Cob length, in cm.
- **CD** Cob diameter, in mm.
- **CW** Cob weight, in g.
- **KW** Kernel weight, in cm.
- **NR** Number of rows.
- **NKR** Number of kernels per row.
- **CDED** Cob diameter / Ear diameter ratio.
- **PERK** Percentage of kernels.
- **TKW** Thousand-kernel weight
- **NKE** Number of kernels per row.

Author(s)

Tiago Olivoto <tiagoolivoto@gmail.com>

Source

Personal data

desc_stat *Descriptive statistics*

Description

Compute the most used measures of central tendency, position, and dispersion.

Usage

```
desc_stat(
  .data = NULL,
  ...,
  by = NULL,
  values = NULL,
  stats = "main",
  hist = FALSE,
  level = 0.95,
  digits = 4,
  na.rm = FALSE,
  verbose = TRUE,
  plot_theme = theme_metan()
)
```

Arguments

.data	The data to be analyzed. Must be a dataframe or an object of class split_factors.
...	A single variable name or a comma-separated list of unquoted variables names. If no variable is informed, all the numeric from .data variables will be used.
by	One variable (factor) to split the data into subsets. The function is then applied to each subset and returns a list where each element contains the results for one level of the variable in by. To split the data by more than one factor variable, use the function split_factors to pass subsetted data to .data.
values	An alternative way to pass the data to the function. It must be a numeric vector.
stats	The descriptive statistics to show. Defaults to "main" (main statistics). Set to "all" to compute all the statistics bellow or chose one (or more) of the following: 'AV.dev' (average deviation), 'CI.mean' (confidence interval for the mean), 'CV' (coefficient of variation), 'IQR' (interquartile range), 'gm.mean' (geometric mean), 'hm.mean' (harmonic mean), 'Kurt' (kurtosis), 'mad' (median absolute deviation), 'max' (maximum value), 'mean' (arithmetic mean), 'median' (median), 'min' (minimum value), 'n' (the length of the data), 'Q2.5' (the percentile 2.5%), 'Q25' (the first quartile, Q1), 'Q75' (the third quartile, Q3), 'Q97.5' (the percentile 97.5%), range (The range of data), 'SD.amo' (the sample standard deviation), 'SD.pop' (the population standard deviation), 'SE.mean' (the standard error of the mean), 'skew' (the skewness), sum (the sum of the values), sum.dev (the sum of the absolute deviations), sum.sq.dev (the sum of the squared deviations), valid.n (The size of sample with valid

	number (not NA), 'var.amo' (the sample variance), 'var.pop' (the population variance). Use a comma-separated vector of names to select the statistics. For example, stats = c("median,mean,CV,n")
hist	Logical argument defaults to FALSE. If hist = TRUE then a histogram is created for each selected variable.
level	The confidence level to compute the confidence interval of mean. Defaults to 0.95.
digits	The number of significant digits.
na.rm	Logical. Should missing values be removed?
verbose	Logical argument. If verbose = FALSE the code is run silently.
plot_theme	The graphical theme of the plot. Default is plot_theme = theme_metan(). For more details, see theme.

Details

In cases when the statistics are computed for more than two variables with data coming from the function split_factors to pass subsetted data.to pass subsetted data to code.data.to pass subsetted data to code.data.the results are returned in a *long* format. Thus, use the function desc_wider to convert it into a *wide* format (levels of the factors in the rows and statistics in the columns).

Value

A tibble with the statistics in the lines and variables in columns. If .data is an object of class split_factors, then the statistics will be shown for each level of the grouping variable in the function split_factors to pass subsetted data.to pass subsetted data to code.data.to pass subsetted data to code.data.

Author(s)

Tiago Olivoto <tiagoolivoto@gmail.com>

Examples

```
library(metan)

desc_stat(data_ge2, TKW)

# Compute the main statistics
# Use a numeric vector as input data
vect <- data_ge2$TKW
desc_stat(values = vect)

# Select specific statistics
desc_stat(values = c(12, 13, 19, 21, 8, NA, 23, NA),
          na.rm = TRUE,
          stats = c('mean, se.mean, cv, n, valid.n'))

# Compute the main statistics for each level of "ENV"
```

```
stats <-
  desc_stat(data_ge2,
            EP, EL, EH, ED, PH, CD,
            by = ENV,
            verbose = FALSE)

# To get a 'wide' format with the statistics of the variable EP above.
desc_wider(stats, PH)

# Compute all the statistics for each combination of "ENV" and "GEN"
# All the numeric variables in .data

stats_all <-
  data_ge2 %>%
  split_factors(ENV, GEN) %>%
  desc_stat(stats = "all", verbose = FALSE)
desc_wider(stats_all, PH)
```

`desc_wider` *Descriptive statistics from long to wide*

Description

`desc_wider` 'widens' an object of class `desc_stat` increasing the number of columns and decreasing the number of rows. This is speccialy useful when the descriptive statistics were computed for each level of a factor using the function `split_factors` to pass subsetted data.to pass subsetted data to code.data.to pass subsetted data to code.data.

Usage

```
desc_wider(.data, var)
```

Arguments

`.data`	An output of the function `desc_stat`.
`var`	The variable in `.data` to show the results.

Value

A tibble with the **statistics in the columns** and **levels of the factor(s) in the rows**.

Author(s)

Tiago Olivoto <tiagoolivoto@gmail.com>

Examples

```
library(metan)

df <-
 data_ge2 %>%
 split_factors(GEN) %>%
 desc_stat(EP, EL, PH, CL,
           stats = c('mean, CI.mean, SE.mean, var.amo, CV'),
           verbose = FALSE)

print(df, n = 15)
desc_wider(df, PH)
```

---

ecovalence *Stability analysis based on Wricke's model*

Description

The function computes the ecovalence (Wricke, 1965) for stability analysis.

Usage

```
ecovalence(.data, env, gen, rep, resp, verbose = TRUE)
```

Arguments

.data	The dataset containing the columns related to Environments, Genotypes, replication/block and response variable(s).
env	The name of the column that contains the levels of the environments.
gen	The name of the column that contains the levels of the genotypes.
rep	The name of the column that contains the levels of the replications/blocks.
resp	The response variable(s). To analyze multiple variables in a single procedure use, for example, resp = c(var1,var2,var3).
verbose	Logical argument. If verbose = FALSE the code will run silently.

Value

An object of class ecovalence containing the results for each variable used in the argument resp.

Author(s)

Tiago Olivoto <tiagoolivoto@gmail.com>

References

Wricke, G. 1965. Zur berechnung der okovalenz bei sommerweizen und hafer. Z. Pflanzenzuchtg 52:127-138.

Examples

```
library(metan)
out <- ecovalence(data_ge2,
                  env = ENV,
                  gen = GEN,
                  rep = REP,
                  resp = PH)
```

---

env_dissimilarity *Dissimilarity between environments*

Description

Computes the dissimilarity between environments based on several approaches. See the section **details** for more details.

Usage

```
env_dissimilarity(.data, env, gen, rep, resp)
```

Arguments

.data	The dataset containing the columns related to Environments, Genotypes, replication/block and response variable(s).
env	The name of the column that contains the levels of the environments.
gen	The name of the column that contains the levels of the genotypes.
rep	The name of the column that contains the levels of the replications/blocks.
resp	The response variable(s). To analyze multiple variables in a single procedure a vector of variables may be used. For example `resp = c(var1,var2,var3)`. Select helpers are also allowed.

Details

Roberteson (1959) proposed the partition of the mean square of the genotype-environment interaction (MS_GE) into single (S) and complex (C) parts, where $S = \frac{1}{2}(\sqrt{Q1} - \sqrt{Q2})^2)$ and $C = (1-r)\sqrt{Q1 - Q2}$, being *r* the correlation between the genotype's average in the two environments; and *Q1* and *Q2* the genotype mean square in the environments 1 and 2, respectively. Cruz and Castoldi (1991) proposed a new decomposition of the MS_GE, in which the complex part is given by $C = \sqrt{(1-r)^3 \times Q1 \times Q2}$.

Value

A list with the following matrices:

- SPART_CC: The percentage of the single (non cross-over) part of the interaction between genotypes and pairs of environments according to the method proposed by Cruz and Castoldi (1991).
- CPART_CC: The percentage of the complex (cross-over) part of the interaction between genotypes and pairs of environments according to the method proposed by Cruz and Castoldi (1991).
- SPART_RO: The percentage of the single (non cross-over) part of the interaction between genotypes and pairs of environments according to the method proposed by Robertson (1959).
- CPART_RO: The percentage of the complex (cross-over) part of the interaction between genotypes and pairs of environments according to the method proposed by Robertson (1959).
- MSGE: Interaction mean square between genotypes and pairs of environments.
- SSGE: Interaction sum of square between genotypes and pairs of environments.
- correlation: Correlation coefficients between genotypes's average in each pair of environment.

Author(s)

Tiago Olivoto <tiagoolivoto@gmail.com>

References

Cruz, C.D., Castoldi, F. (1991). Decomposicao da interacao genotipos x ambientes em partes simples e complexa. Ceres, 38:422-430. Available at: http://www.ceres.ufv.br/ojs/index.php/ceres/article/view/2165.

Robertson, A. (1959). Experimental design on the measurement of heritabilities and genetic correlations. biometrical genetics. New York: Pergamon Press.

Examples

```
mod <- env_dissimilarity(data_ge, ENV, GEN, REP, GY)
print(mod)
```

---

fai_blup *Multi-trait selection index*

Description

Multitrait index based on factor analysis and ideotype-design proposed by Rocha et al. (2018).

Usage

```
fai_blup(.data, DI, UI, SI = NULL, mineval = 1, verbose = TRUE)
```

Arguments

.data	An object of class waasb or a two-way table with genotypes in the rows and traits in columns. In the last case the row names must contain the genotypes names.
DI, UI	A vector of the same length of .data to construct the desirable (DI) and undesirable (UI) ideotypes. For each element of the vector, allowed values are 'max', 'min', 'mean', or a numeric value. Use a comma-separated vector of text. For example, DI = c("max,max,min,min").
SI	An integer (0-100). The selection intensity in percentage of the total number of genotypes.
mineval	The minimum value so that an eigenvector is retained in the factor analysis.
verbose	Logical value. If TRUE some results are shown in console.

Value

An object of class fai_blup with the following items:

- **data** The data (BLUPS) used to compute the index.
- **FA** The results of the factor analysis.
- **canonical.loadings** The canonical loadings for each factor retained.
- **FAI** A list with the FAI-BLUP index for each ideotype design.
- **selection.diferential** A list with the selection differential for each ideotype design.

Author(s)

Tiago Olivoto <tiagoolivoto@gmail.com>

References

Rocha, J.R.A.S.C.R, J.C. Machado, and P.C.S. Carneiro. 2018. Multitrait index based on factor analysis and ideotype-design: proposal and application on elephant grass breeding for bioenergy. GCB Bioenergy 10:52-60. doi: doi:10.1111/gcbb.12443

Examples

```
library(metan)

mod <- waasb(data_ge,
             env = ENV,
             gen = GEN,
             rep = REP,
             resp = c(GY, HM))

FAI <- fai_blup(mod,
                SI = 15,
                DI = c('max, max'),
                UI = c('min, min'))
```

```
# Or using the pipe operator %>%

FAI <- data_ge2 %>%
       waasb(ENV, GEN, REP, c(KW, NKE, PH, EH)) %>%
       fai_blup(DI = c('max, max, max, min'),
                UI = c('min, min, min, max'),
                SI = 15)
```

`find_outliers` *Find possible outliers in a dataset*

Description

Find possible outliers in the dataset.

Usage

```
find_outliers(
  .data = NULL,
  var = NULL,
  by = NULL,
  values = NULL,
  plots = FALSE,
  coef = 1.5,
  verbose = TRUE,
  plot_theme = theme_metan()
)
```

Arguments

`.data`	The data to be analyzed. Must be a dataframe or an object of class `split_factors`.
`var`	The variable to be analyzed.
`by`	One variable (factor) to split the data into subsets. The function is then applied to each subset and returns a list where each element contains the results for one level of the variable in by. To split the data by more than one factor variable, use the function `split_factors` to pass subsetted data to `.data`.
`values`	An alternative way to pass the data to the function. It must be a numeric vector.
`plots`	If `TRUE`, then histograms and boxplots are shown.
`coef`	The multiplication coefficient, defaults to 1.5. For more details see ?`boxplot.stat`.
`verbose`	If `verbose = TRUE` then some results are shown in the console.
`plot_theme`	The graphical theme of the plot. Default is `plot_theme = theme_metan()`. For more details, see `theme`.

Author(s)

Tiago Olivoto <tiagoolivoto@gmail.com>

Examples

```
library(metan)

find_outliers(data_ge2, var = PH, plots = TRUE)

# Find outliers within each environment
find_outliers(data_ge2, var = PH, by = ENV)
```

Fox *Fox's stability function*

Description

Performs a stability analysis based on the criteria of Fox et al. (1990), using the statistical "TOP third" only. A stratified ranking of the genotypes at each environment is done. The proportion of locations at which the genotype occurred in the top third are expressed in the output.

Usage

```
Fox(.data, env, gen, rep, resp, verbose = TRUE)
```

Arguments

.data	The dataset containing the columns related to Environments, Genotypes, replication/block and response variable(s).
env	The name of the column that contains the levels of the environments.
gen	The name of the column that contains the levels of the genotypes.
rep	The name of the column that contains the levels of the replications/blocks.
resp	The response variable(s). To analyze multiple variables in a single procedure use, for example, resp = c(var1,var2,var3).
verbose	Logical argument. If verbose = FALSE the code will run silently.

Value

An object of class Fox, which is a list containing the results for each variable used in the argument resp. For each variable, a tibble with the following columns is returned.

- **GEN** the genotype's code.
- **mean** the mean for the response variable.
- **TOP** The proportion of locations at which the

Author(s)

Tiago Olivoto <tiagoolivoto@gmail.com>

References

Fox, P.N., B. Skovmand, B.K. Thompson, H.J. Braun, and R. Cormier. 1990. Yield and adaptation of hexaploid spring triticale. Euphytica 47:57-64. doi:10.1007/BF00040364.

Examples

```
library(metan)
out = Fox(data_ge2,
          env = ENV,
          gen = GEN,
          rep = REP,
          resp = PH)
```

---

gai *Geometric adaptability index*

Description

Performs a stability analysis based on the geometric mean (GAI), according to the following model:

$$GAI = \sqrt[E]{\bar{Y}_{1.} \cdot \bar{Y}_{2.} \cdot ... \cdot \bar{Y}_{i.}}$$

Usage

```
gai(.data, env, gen, rep, resp, verbose = TRUE)
```

Arguments

.data	The dataset containing the columns related to Environments, Genotypes, replication/block and response variable(s).
env	The name of the column that contains the levels of the environments.
gen	The name of the column that contains the levels of the genotypes.
rep	The name of the column that contains the levels of the replications/blocks.
resp	The response variable(s). To analyze multiple variables in a single procedure use, for example, resp = c(var1,var2,var3).
verbose	Logical argument. If verbose = FALSE the code will run silently.

Value

An object of class `gai`, which is a list containing the results for each variable used in the argument `resp`. For each variable, a tibble with the following columns is returned.

- **GEN** the genotype's code.
- **GAI** Geometric adaptability index
- **GAI_R** The rank for the GAI value.

Author(s)

Tiago Olivoto <tiagoolivoto@gmail.com>

References

Shahbazi, E. 2019. Genotype selection and stability analysis for seed yield of Nigella sativa using parametric and non-parametric statistics. Sci. Hortic. (Amsterdam). 253:172-179. doi:10.1016/j.scienta.2019.04.047.

Examples

```
library(metan)
out <- gai(data_ge2, ENV, GEN, REP, c(EH, PH, EL, CD, ED, NKE))
```

---

gamem *Genotype analysis by mixed-effect models*

Description

Analysis of genotypes in single experiments using mixed-effect models with estimation of genetic parameters.

Usage

```
gamem(.data, gen, rep, resp, block = NULL, prob = 0.05, verbose = TRUE)
```

Arguments

`.data`	The dataset containing the columns related to, Genotypes, replication/block and response variable(s).
`gen`	The name of the column that contains the levels of the genotypes, that will be treated as random effect.
`rep`	The name of the column that contains the levels of the replications (assumed to be fixed).
`resp`	The response variable(s). To analyze multiple variables in a single procedure a vector of variables may be used. For example `resp = c(var1,var2,var3)`. Select helpers are also allowed.

block	Defaults to NULL. In this case, a randomized complete block design is considered. If block is informed, then an alpha-lattice design is employed considering block as random to make use of inter-block information, whereas the complete replicate effect is always taken as fixed, as no inter-replicate information was to be recovered (Mohring et al., 2015).
prob	The probability for estimating confidence interval for BLUP's prediction.
verbose	Logical argument. If verbose = FALSE the code are run silently.

Details

gamem analyses data from a one-way genotype testing experiment. By default, a randomized complete block design is used according to the following model:

$$Y_{ij} = m + g_i + r_j + e_{ij}$$

where Y_{ij} is the response variable of the ith genotype in the *j*th block; *m* is the grand mean (fixed); g_i is the effect of the *i*th genotype (assumed to be random); r_j is the effect of the *j*th replicate (assumed to be fixed); and e_{ij} is the random error.

When block is informed, then a resolvable alpha design is implemented, according to the following model:

$$Y_{ijk} = m + g_i + r_j + b_{jk} + e_{ijk}$$

where where y_{ijk} is the response variable of the *i*th genotype in the *k*th block of the *j*th replicate; *m* is the intercept, t_i is the effect for the *i*th genotype r_j is the effect of the *j*th replicate, b_{jk} is the effect of the *k*th incomplete block of the *j*th replicate, and e_{ijk} is the plot error effect corresponding to y_{ijk}.

Value

An object of class gamem, which is a list with the following items for each element (variable):

- **fixed:** Test for fixed effects.
- **random:** Variance components for random effects.
- **LRT:** The Likelihood Ratio Test for the random effects.
- **blupGEN:** The estimated BLUPS for genotypes
- **Details:** A tibble with the following data: Ngen, the number of genotypes; OVmean, the grand mean; Min, the minimum observed (returning the genotype and replication/block); Max the maximum observed, MinGEN the winner genotype, MaxGEN, the loser genotype.
- **ESTIMATES:** A tibble with the values for the genotypic variance, block-within-replicate variance (if an alpha-lattice design is used by informing the block in block), the residual variance and their respective contribution to the phenotypic variance; broad-sence heritability, heritability on the entry-mean basis, genotypic coefficient of variation residual coefficient of variation and ratio between genotypic and residual coefficient of variation.
- **residuals:** The residuals of the model.

Author(s)

Tiago Olivoto <tiagoolivoto@gmail.com>

References

Mohring, J., E. Williams, and H.-P. Piepho. 2015. Inter-block information: to recover or not to recover it? TAG. Theor. Appl. Genet. 128:1541-54. doi:10.1007/s00122-015-2530-0

See Also

get_model_data waasb

Examples

```
library(metan)

# fitting the model considering an RCBD
# Genotype as random effects

rcbd <- gamem(data_g,
              gen = GEN,
              rep = REP,
              resp = c(PH, ED, EL, CL, CW, KW, NR, TKW, NKE))

# Likelihood ratio test for random effects
## Statistic
get_model_data(rcbd, "lrt")

## P-value
get_model_data(rcbd, "pval_lrt")

# Variance components
get_model_data(rcbd, "vcomp")

# Genetic parameters
get_model_data(rcbd, "genpar")

# BLUPs for genotypes
get_model_data(rcbd)

# fitting the model considering an alpha-lattice design
# Genotype and block-within-replicate as random effects
# Note that block effect was now informed.

alpha <- gamem(data_alpha,
               gen = GEN,
               rep = REP,
               block = BLOCK,
               resp = YIELD)
# Use the function  get_model_data() to easely extract the model values.
get_model_data(alpha, "genpar")
```

`get_model_data` *Get data from a model easily*

Description

Easily get data from some objects generated in the **metan** package such as the WAASB and WAASBY indexes (Olivoto et al., 2019a, 2019b) BLUPs, variance components, details of AMMI models and AMMI-based stability statistics.

Usage

```
get_model_data(x, what = NULL, type = "GEN")
```

Arguments

`x`	An object created with the functions `AMMI_indexes`, `ecovalence`, `Fox`, `gai`, `gamem`, `ge_means`, `ge_reg`, `performs_ammi`, `Resende_indexes`, `Shukla`, `superiority`, `waas` or `waasb`.
`what`	What should be captured from the model. See more in section **Details**.
`type`	Chose if the statistics must be show by genotype (`type = "GEN"`, default) or environment (`type = "ENV"`), when possible.

Details

Bellow are listed the options allowed in the argument `what` depending on the class of the object

Objects of class `AMMI_indexes`**:**

- `"ASV"` AMMI stability value.
- `"EV"` Averages of the squared eigenvector values.
- `"SIPC"` Sums of the absolute value of the IPCA scores.
- `"WAAS"` Weighted average of absolute scores (default).
- `"ZA"` Absolute value of the relative contribution of IPCAs to the interaction.

Objects of class `Annicchiarico` **and** `Schmildt`**:**

- `"Sem_rp"` The standard error of the relative mean performance (Schmildt).
- `"Mean_rp"` The relative performance of the mean.
- `"rank"` The rank for genotypic confidence index.
- `"Wi"` The genotypic confidence index.

Objects of class `ecovalence`**:**

- `"Ecoval"` Ecovalence value (default).

- "Ecov_perc" Ecovalence in percentage value.
- "rank" Rank for ecovalence.

Objects of class ge_reg**:**

- "deviations" The deviations from regression.
- "RMSE" The Root Mean Square Error.
- "R2" The r-square of the regression.
- "slope" The sloop of the regression (default).

Objects of class ge_effects**:**

- For objects of class ge_effects no argument what is required.

Objects of class ge_means**:**

- "ge_means" Genotype-environment interaction means (default).
- "env_means" Environment means.
- "gen_means" Genotype means.

Objects of class Shukla**:**

- "rMean" Rank for the mean.
- "ShuklaVar" Shukla's stablity variance (default).
- "rShukaVar" Rank for Shukla's stablity variance.
- "ssiShukaVar" Simultaneous selection index.

Objects of class Fox**:**

- "TOP" The proportion of locations at which the genotype occurred in the top third (default).

Objects of class gai**:**

- "GAI" The geometric adaptability index (default).
- "GAI_R" The rank for the GAI values.

Objects of class superiority**:**

- "Pi_a" The superiority measure for all environments (default).
- "R_a" The rank for Pi_a.
- "Pi_f" The superiority measure for favorable environments.
- "R_f" The rank for Pi_f.
- "Pi_u" The superiority measure for unfavorable environments.
- "R_u" The rank for Pi_u.

Objects of class Huehn**:**

- "S1" Mean of the absolute rank differences of a genotype over the n environments (default).
- "S2" variance among the ranks over the k environments.

- "S3" Sum of the absolute deviations.
- "S6" Relative sum of squares of rank for each genotype.
- "S1_R", "S2_R", "S3_R", and "S6_R", the ranks for S1, S2, S3, and S6, respectively.

Objects of class Thennarasu**:**

- "N1" First statistic (default).
- "N2" Second statistic.
- "N3" Third statistic.
- "N4" Fourth statistic.
- "N1_R", "N2_R", "N3_R", and "N4_R", The ranks for the statistics.

Objects of class performs_ammi**:**

- "PC1","PC2",...,"PCn" The values for the nth interaction principal component axis.
- "ipca_ss" Sum of square for each IPCA.
- "ipca_ms" Mean square for each IPCA.
- "ipca_fval" F value for each IPCA.
- "ipca_pval" P-value for for each IPCA.
- "ipca_expl" Explained sum of square for each IPCA (default).
- "ipca_accum" Accumulated explained sum of square.

Objects of class waas**,** waas_means**, and** waasb**:**

- "PC1","PC2",...,"PCn" The values for the nth interaction principal component axis.
- "WAASB" The weighted average of the absolute scores (default for objects of class waas).
- "PctResp" The rescaled values of the response variable.
- "PctWAASB" The rescaled values of the WAASB.
- "wResp" The weight for the response variable.
- "wWAASB" The weight for the stability.
- "OrResp" The ranking regarding the response variable.
- "OrWAASB" The ranking regarding the WAASB.
- "OrPC1" The ranking regarding the first principal component axix.
- "WAASBY" The superiority index WAASBY.
- "OrWAASBY" The ranking regarding the superiority index.

Objects of class waasb **or** gamem**:**

- "blupg" For genotype's predicted mean.
- "blupge" for genotype-vs-environment's predicted mean (only for objects of class waasb).
- "genpar" Genetic parameters (default).
- "lrt" The statistic for the likelihood-ratio test for random effects.
- "pval_lrt" The p-values for the likelihood-ratio test.

- "vcomp" The variance components for random effects.

Objects of class Res_ind

- "HMGV" For harmonic mean of genotypic values.
- "RPGV or RPGV_Y" For relative performance of genotypic values
- "HMRPGV" For harmonic mean of relative performance of genotypic values

Value

A tibble showing the values of the variable chosen in argument what.

Author(s)

Tiago Olivoto <tiagoolivoto@gmail.com>

References

Annicchiarico, P. 1992. Cultivar adaptation and recommendation from alfalfa trials in Northern Italy. J. Genet. Breed. 46:269-278.

Dias, P.C., A. Xavier, M.D.V. de Resende, M.H.P. Barbosa, F.A. Biernaski, R.A. Estopa. 2018. Genetic evaluation of Pinus taeda clones from somatic embryogenesis and their genotype x environment interaction. Crop Breed. Appl. Biotechnol. 18:55-64. doi:10.1590/1984-70332018v18n1a8

Azevedo Peixoto, L. de, P.E. Teodoro, L.A. Silva, E.V. Rodrigues, B.G. Laviola, and L.L. Bhering. 2018. Jatropha half-sib family selection with high adaptability and genotypic stability. PLoS One 13:e0199880. doi:10.1371/journal.pone.0199880

Eberhart, S.A., and W.A. Russell. 1966. Stability parameters for comparing Varieties. Crop Sci. 6:36-40. doi:10.2135/cropsci1966.0011183X000600010011x.

Fox, P.N., B. Skovmand, B.K. Thompson, H.J. Braun, and R. Cormier. 1990. Yield and adaptation of hexaploid spring triticale. Euphytica 47:57-64. doi:10.1007/BF00040364.

Huehn, V.M. 1979. Beitrage zur erfassung der phanotypischen stabilitat. EDV Med. Biol. 10:112.

Olivoto, T., A.D.C. L\'ucio, J.A.G. da silva, V.S. Marchioro, V.Q. de Souza, and E. Jost. 2019a. Mean performance and stability in multi-environment trials I: Combining features of AMMI and BLUP techniques. Agron. J. 111:2949-2960. doi:10.2134/agronj2019.03.0220

Olivoto, T., A.D.C. L\'ucio, J.A.G. da silva, B.G. Sari, and M.I. Diel. 2019b. Mean performance and stability in multi-environment trials II: Selection based on multiple traits. Agron. J. 111:2961-2969. doi:10.2134/agronj2019.03.0221

Purchase, J.L., H. Hatting, and C.S. van Deventer. 2000. Genotype vs environment interaction of winter wheat (Triticum aestivum L.) in South Africa: II. Stability analysis of yield performance. South African J. Plant Soil 17:101-107. doi:10.1080/02571862.2000.10634878

Resende MDV (2007) Matematica e estatistica na analise de experimentos e no melhoramento genetico. Embrapa Florestas, Colombo

Sneller, C.H., L. Kilgore-Norquest, and D. Dombek. 1997. Repeatability of Yield Stability Statistics in Soybean. Crop Sci. 37:383-390. doi:10.2135/cropsci1997.0011183X003700020013x

Shahbazi, E. 2019. Genotype selection and stability analysis for seed yield of Nigella sativa using parametric and non-parametric statistics. Sci. Hortic. (Amsterdam). 253:172-179. doi:10.1016/j.scienta.2019.04.047.

Wricke, G. 1965. Zur berechnung der okovalenz bei sommerweizen und hafer. Z. Pflanzenzuchtg 52:127-138.

Zali, H., E. Farshadfar, S.H. Sabaghpour, and R. Karimizadeh. 2012. Evaluation of genotype vs environment interaction in chickpea using measures of stability from AMMI model. Ann. Biol. Res. 3:3126-3136. http://eprints.icrisat.ac.in/id/eprint/7173

Examples

```
library(metan)

#################### joint-regression analysis ####################
ge_r <- ge_reg(data_ge2, ENV, GEN, REP,
               resp = c(PH, EH, CD, CL, ED))
get_model_data(ge_r)
get_model_data(ge_r, "deviations")


#################### AMMI model ####################
# Fit an AMMI model for 7 variables.
AMMI <- data_ge2 %>%
 performs_ammi(ENV, GEN, REP,
               resp = c(PH, ED, TKW, NKR, CD, CL, CW))

# Sum of squares
get_model_data(AMMI, "ipca_ss")

# Mean squares
get_model_data(AMMI, "ipca_ms")

# Examine the significance (p-value) of the IPCAs
get_model_data(AMMI, "ipca_pval")

# Explained sum of square for each IPCA
get_model_data(AMMI)

# Accumulated sum of square
get_model_data(AMMI, "ipca_accum")

### AMMI-based stability statistics ###
# Get the AMMI stability value
AMMI %>%
AMMI_indexes() %>%
get_model_data("ASV")


#################### WAASB model ####################
# Fitting the WAAS index
AMMI <- waas(data_ge2, ENV, GEN, REP,
             resp = c(PH, ED, TKW, NKR))

# Getting the weighted average of absolute scores
```

```
get_model_data(AMMI, what = "WAASB")

# And the rank for the WAASB index.
get_model_data(AMMI, what = "OrWAASB")


#################### BLUP model ####################
# Fitting a mixed-effect model
blup <- waasb(data_ge2, ENV, GEN, REP,
              resp = c(PH, ED, TKW, NKR))

# Getting p-values for likelihood-ratio test
get_model_data(blup, what = "pval_lrt")

# Getting the variance components
get_model_data(blup, what = "vcomp")

# Getting the genetic parameters
get_model_data(blup)

### BLUP-based stability indexes ###
blup %>%
Resende_indexes() %>%
get_model_data()


#################### Stability indexes ####################
stats_ge <- ge_stats(data_ge, ENV, GEN, REP, everything())
get_model_data(stats_ge)
```

---

ge_cluster *Cluster genotypes or environments*

Description

Performs clustering for genotypes or tester environments based on a dissimilarity matrix.

Usage

```
ge_cluster(
  .data,
  env = NULL,
  gen = NULL,
  resp = NULL,
  table = FALSE,
  distmethod = "euclidean",
  clustmethod = "ward.D",
```

```
  scale = TRUE,
  cluster = "env",
  nclust = NULL
)
```

Arguments

.data	The dataset containing the columns related to Environments, Genotypes and the response variable. It is also possible to use a two-way table with genotypes in lines and environments in columns as input. In this case you must use table = TRUE.
env	The name of the column that contains the levels of the environments. Defaults to NULL, in case of the input data is a two-way table.
gen	The name of the column that contains the levels of the genotypes. Defaults to NULL, in case of the input data is a two-way table.
resp	The response variable(s). Defaults to NULL, in case of the input data is a two-way table.
table	Logical values indicating if the input data is a two-way table with genotypes in the rows and environments in the columns. Defaults to FALSE.
distmethod	The distance measure to be used. This must be one of 'euclidean', 'maximum', 'manhattan', 'canberra', 'binary', or 'minkowski'.
clustmethod	The agglomeration method to be used. This should be one of 'ward.D' (Default), 'ward.D2', 'single', 'complete', 'average' (= UPGMA), 'mcquitty' (= WPGMA), 'median' (= WPGMC) or 'centroid' (= UPGMC).
scale	Should the data be scaled befor computing the distances? Set to TRUE. Let Y_{ij} be the yield of Hybrid *i* in Location *j*, $\bar{Y}_{.j}$ be the mean yield, and S_j be the standard deviation of Location *j*. The standardized yield (Zij) is computed as (Ouyang et al. 1995): $Z_{ij} = (Y_{ij} - Y_{.j})/S_j$.
cluster	What should be clustered? Defaults to cluster = "env" (cluster environments). To cluster the genotypes use cluster = "gen".
nclust	The number of clust to be formed. Set to NULL.

Value

- **data** The data that was used to compute the distances.
- **cutpoint** The cutpoint of the dendrogram according to Mojena (1977).
- **distance** The matrix with the distances.
- **de** The distances in an object of class dist.
- **hc** The hierarchical clustering.
- **cophenetic** The cophenetic correlation coefficient between distance matrix and cophenetic matrix
- **Sqt** The total sum of squares.
- **tab** A table with the clusters and similarity.
- **clusters** The sum of square and the mean of the clusters for each genotype (if cluster = "env" or environment (if cluster = "gen").
- **labclust The labels** of genotypes/environments within each cluster.

Author(s)

Tiago Olivoto <tiagoolivoto@gmail.com>

References

Mojena, R. 2015. Hierarchical grouping methods and stopping rules: an evaluation. Comput. J. 20:359-363. doi:10.1093/comjnl/20.4.359

Ouyang, Z., R.P. Mowers, A. Jensen, S. Wang, and S. Zheng. 1995. Cluster analysis for genotype x environment interaction with unbalanced data. Crop Sci. 35:1300-1305. doi:10.2135/cropsci1995.0011183X003500050008x

Examples

```
library(metan)

d1 = ge_cluster(data_ge, ENV, GEN, GY, nclust = 3)
plot(d1, nclust = 3)
```

---

ge_details *Details for genotype-environment trials*

Description

Details for genotype-environment trials

Usage

```
ge_details(.data, env, gen, resp)
```

Arguments

.data	The dataset containing the columns related to Environments, Genotypes, replication/block and response variable(s).
env	The name of the column that contains the levels of the environments.
gen	The name of the column that contains the levels of the genotypes.
resp	The response variable(s). To analyze multiple variables in a single procedure a vector of variables may be used. For example resp = c(var1,var2,var3). Select helpers are also allowed.

Value

A tibble with the following results for each variable:

- `Mean`: The grand mean.
- `Std_err`: The standard error of the mean
- `Desv_pad`: The standard deviation.
- `CV`: The coefficient of variation.
- `Min,Max`: The minimum and maximum value, indicating the genotype and environment of occurence.
- `MinENV,MinGEN`: The environment and genotype with the lower mean.
- `MaxENV,MaxGEN`: The environment and genotype with the higher mean.

Author(s)

Tiago Olivoto <tiagoolivoto@gmail.com>

Examples

```
library(metan)
details <- ge_details(data_ge2, ENV, GEN, everything())
print(details)
```

---

ge_effects *Genotype-environment effects*

Description

This is a helper function that computes the genotype-environment effects, i.e., the residual effect of the additive model

Usage

```
ge_effects(.data, env, gen, rep, resp, type = "ge", verbose = TRUE)
```

Arguments

`.data`	The dataset containing the columns related to Environments, Genotypes, replication/block and response variable(s).
`env`	The name of the column that contains the levels of the environments. The analysis of variance is computed for each level of this factor.
`gen`	The name of the column that contains the levels of the genotypes.
`rep`	The name of the column that contains the levels of the replications/blocks.

resp	The response variable(s). To analyze multiple variables in a single procedure a vector of variables may be used. For example resp = c(var1,var2,var3).
type	The type of effect to compute. Defaults to "ge", i.e., genotype-environment. To compute genotype plus genotype-environment effects use type = "gge".
verbose	Logical argument. If verbose = FALSE the code will run silently.

Value

A list where each element is the result for one variable that contains a two-way table with genotypes in rows and environments in columns.

Author(s)

Tiago Olivoto <tiagoolivoto@gmail.com>

Examples

```
library(metan)
ge_eff <- ge_effects(data_ge, ENV, GEN, REP, GY)
gge_eff <- ge_effects(data_ge, ENV, GEN, REP, GY, type = "gge")
plot(ge_eff)
```

---

ge_factanal *Stability analysis and environment stratification*

Description

This function computes the stability analysis and environmental stratification using factor analysis as proposed by Murakami and Cruz (2004).

Usage

```
ge_factanal(.data, env, gen, rep, resp, mineval = 1, verbose = TRUE)
```

Arguments

.data	The dataset containing the columns related to Environments, Genotypes, replication/block and response variable(s)
env	The name of the column that contains the levels of the environments.
gen	The name of the column that contains the levels of the genotypes.
rep	The name of the column that contains the levels of the replications/blocks
resp	The response variable(s). To analyze multiple variables in a single procedure use, for example, resp = c(var1,var2,var3).
mineval	The minimum value so that an eigenvector is retained in the factor analysis.
verbose	Logical argument. If verbose = FALSE the code will run silently.

Value

An object of class `ge_factanal` with the following items:

`data`	The data used to compute the factor analysis.
`cormat`	The correlation matrix among the environments.
`PCA`	The eigenvalues and explained variance.
`FA`	The factor analysis.
`env_strat`	The environmental stratification.
`KMO`	The result for the Kaiser-Meyer-Olkin test.
`MSA`	The measure of sampling adequacy for individual variable.
`communalities`	The communalities.
`communalities.mean`	The communalities' mean.
`initial.loadings`	The initial loadings.
`finish.loadings`	The final loadings after varimax rotation.
`canonical.loadings`	The canonical loadings.
`scores.gen`	The scores for genotypes for the first and second factors.

Author(s)

Tiago Olivoto, <tiagoolivoto@gmail.com>

References

Murakami, D.M.D., and C.D.C. Cruz. 2004. Proposal of methodologies for environment stratification and analysis of genotype adaptability. Crop Breed. Appl. Biotechnol. 4:7-11.

See Also

`superiority`, `ecovalence`, `ge_stats`, `ge_reg`

Examples

```
library(metan)
model = ge_factanal(data_ge2,
                    env = ENV,
                    gen = GEN,
                    rep = REP,
                    resp = PH)
```

`ge_means` *Genotype-environment means*

Description

Computes genotype-environment interaction means

Usage

```
ge_means(.data, env, gen, resp)
```

Arguments

`.data`	The dataset containing the columns related to Environments, Genotypes, and the response variable(s).
`env`	The name of the column that contains the levels of the environments.
`gen`	The name of the column that contains the levels of the genotypes.
`resp`	The response variable(s). To analyze multiple variables at once, a vector of variables may be used. For example `resp = c(var1,var2,var3)`. Select helpers are also allowed.

Value

A list where each element is the result for one variable containing:

- **ge_means**: A two-way table with the means for genotypes (rows) and environments (columns).
- **gen_means**: A tibble with the means for genotypes.
- **env_means**: A tibble with the means for environments.

Author(s)

Tiago Olivoto <tiagoolivoto@gmail.com>

Examples

```
library(metan)
means_ge <- ge_means(data_ge, ENV, GEN, resp = everything())

# Genotype-environment interaction means
get_model_data(means_ge)

# Environment means
get_model_data(means_ge, what = "env_means")

# Genotype means
get_model_data(means_ge, what = "gen_means")
```

`ge_plot` *Graphical analysis of genotype-vs-environment interaction*

Description

This function produces a line plot for a graphical interpretation of the genotype-vs-environment interaction. By default, environments are in the x axis whereas the genotypes are depicted by different lines. The y axis contains the value of the selected variable. A heatmap can also be created.

Usage

```
ge_plot(
  .data,
  env,
  gen,
  resp,
  type = 1,
  plot_theme = theme_metan(),
  colour = TRUE
)
```

Arguments

`.data`	The dataset containing the columns related to Environments, Genotypes, replication/block and response variable(s).
`env`	The name of the column that contains the levels of the environments
`gen`	The name of the column that contains the levels of the genotypes.
`resp`	The response variable.
`type`	The type of plot `type = 1` for a heatmap or `type = 2` for a line plot.
`plot_theme`	The graphical theme of the plot. Default is `plot_theme = theme_metan()`. For more details,see `theme`.
`colour`	Logical argument. If `FALSE` then the plot will not be colored.

Value

An object of class `gg,ggplot`.

Author(s)

Tiago Olivoto <tiagoolivoto@gmail.com>

Examples

```
library(metan)
library(ggplot2)
ge_plot(data_ge2, ENV, GEN, PH)
ge_plot(data_ge, ENV, GEN, GY, type = 2)
```

---

ge_reg *Eberhart and Russell's regression model*

Description

Regression-based stability analysis using the Eberhart and Russell (1966) model.

Usage

```
ge_reg(.data, env, gen, rep, resp, verbose = TRUE)
```

Arguments

.data	The dataset containing the columns related to Environments, Genotypes, replication/block and response variable(s)
env	The name of the column that contains the levels of the environments.
gen	The name of the column that contains the levels of the genotypes.
rep	The name of the column that contains the levels of the replications/blocks
resp	The response variable(s). To analyze multiple variables in a single procedure use, for example, resp = c(var1,var2,var3).
verbose	Logical argument. If verbose = FALSE the code will run silently.

Value

An object of class ge_reg with the folloing items for each variable:

data	The data with means for genotype and environment combinations and the environment index
anova	The analysis of variance for the regression model.
regression	The estimated coefficients of the regression model.

Author(s)

Tiago Olivoto, <tiagoolivoto@gmail.com>

References

Eberhart, S.A., and W.A. Russell. 1966. Stability parameters for comparing Varieties. Crop Sci. 6:36-40. doi:10.2135/cropsci1966.0011183X000600010011x.

See Also

superiority,ecovalence,ge_stats

Examples

```
library(metan)
reg <- ge_reg(data_ge2,
              env = ENV,
              gen = GEN,
              rep = REP,
              resp = PH)
plot(reg)
```

---

ge_stats *Statistics for genotype-vs-environment interaction*

Description

Computes **(i)** within-environment analysis of variance, GEI effect, GEI means, and genotype plus GEI effects; **(ii)** parametric statistics including AMMI-based indexes, Annicchiarico's genotypic confidence index (1992), Ecovalence (Wricke, 1965), regression-based stability (Eberhart and Russell., 1966), Shukla's stability variance parameter (1972); and **(iii)** nonparametric statistics including Fox's stability function (Fox et al. 1990), superiority index (Lin and Binns, 1988), Huehn's stability statistics (Huehn, 1979), and Thennarasu (1995) statistics.

Usage

```
ge_stats(.data, env, gen, rep, resp, verbose = TRUE, prob = 0.05)
```

Arguments

.data	The dataset containing the columns related to Environments, Genotypes, replication/block and response variable(s).
env	The name of the column that contains the levels of the environments.
gen	The name of the column that contains the levels of the genotypes.
rep	The name of the column that contains the levels of the replications/blocks.
resp	The response variable(s). To analyze multiple variables in a single procedure use, for example, resp = c(var1,var2,var3).
verbose	Logical argument. If verbose = FALSE the code will run silently.
prob	The probability error assumed.

Details

The function computes the statistics and ranks for the following stability indexes. "Y" (Response variable), "CV" (coefficient of variation), "Var" (Genotype's variance), "Shukla" (Shukla's variance, calling Shukla internally), "Wi_g","Wi_f","Wi_u" (Annichiarrico's genotypic confidence index for all, favorable and unfavorable environments, respectively, calling Annicchiarico internally), "Ecoval" (Wricke's ecovalence, ecovalence internally), "Sij" (Deviations from the joint-regression analysis) and "R2" (R-squared from the joint-regression analysis, calling ge_reg internally), "ASV" (AMMI-stability value), "SIPC" (sum of the absolute values of the IPCA scores), "EV" (Average of the squared eigenvector values), "ZA" (Absolute values of the relative contributions of the IPCAs to the interaction), and "WAAS" (Weighted Average of Absolute Scores), by calling AMMI_indexes internally; "HMGV" (Harmonic mean of the genotypic value), "RPGV" (Relative performance of the genotypic values), "HMRPGV" (Harmonic mean of the relative performance of the genotypic values), by calling Resende_indexes internally; "Pi_a","Pi_f","Pi_u" (Superiority indexes for all, favorable and unfavorable environments, respectively, calling superiority internally), "Gai" (Geometric adaptability index, calling gai internally), "S1" (mean of the absolute rank differences of a genotype over the n environments), "S2" (variance among the ranks over the k environments), "S3" (sum of the absolute deviations), "S6" (relative sum of squares of rank for each genotype), by calling Huehn internally; and "N1","N2","N3","N4" (Thennarasu"s statistics, calling Thennarasu internally).

Value

An object of class ge_stats which is a list with one data frame for each variable containing the computed indexes.

Author(s)

Tiago Olivoto <tiagoolivoto@gmail.com>

References

Annicchiarico, P. 1992. Cultivar adaptation and recommendation from alfalfa trials in Northern Italy. Journal of Genetic \& Breeding, 46:269-278

Eberhart, S.A., and W.A. Russell. 1966. Stability parameters for comparing Varieties. Crop Sci. 6:36-40. doi:10.2135/cropsci1966.0011183X000600010011x

Fox, P.N., B. Skovmand, B.K. Thompson, H.J. Braun, and R. Cormier. 1990. Yield and adaptation of hexaploid spring triticale. Euphytica 47:57-64. doi:10.1007/BF00040364.

Huehn, V.M. 1979. Beitrage zur erfassung der phanotypischen stabilitat. EDV Med. Biol. 10:112.

Kang, M.S., and H.N. Pham. 1991. Simultaneous Selection for High Yielding and Stable Crop Genotypes. Agron. J. 83:161. doi:10.2134/agronj1991.00021962008300010037x.

Lin, C.S., and M.R. Binns. 1988. A superiority measure of cultivar performance for cultivar x location data. Can. J. Plant Sci. 68:193-198. doi:10.4141/cjps88-018

Olivoto, T., A.D.C. L\'ucio, J.A.G. da silva, V.S. Marchioro, V.Q. de Souza, and E. Jost. 2019a. Mean performance and stability in multi-environment trials I: Combining features of AMMI and BLUP techniques. Agron. J. 111:2949-2960. doi:10.2134/agronj2019.03.0220

Shahbazi, E. 2019. Genotype selection and stability analysis for seed yield of Nigella sativa using parametric and non-parametric statistics. Sci. Hortic. (Amsterdam). 253:172-179. doi:10.1016/j.scienta.2019.04.047.

Shukla, G.K. 1972. Some statistical aspects of partitioning genotype-environmental components of variability. Heredity. 29:238-245. doi:10.1038/hdy.1972.87.

Thennarasu, K. 1995. On certain nonparametric procedures for studying genotype x environment interactions and yield stability. Ph.D. thesis. P.J. School, IARI, New Delhi, India.

Wricke, G. 1965. Zur berechnung der okovalenz bei sommerweizen und hafer. Z. Pflanzenzuchtg 52:127-138.

Examples

```
library(metan)

model <- ge_stats(data_ge, ENV, GEN, REP, GY)
get_model_data(model, "stats")
```

`ge_winners` *Genotype-environment winners*

Description

Computes the ranking for genotypes within environments and return the winners.

Usage

```
ge_winners(.data, env, gen, resp, type = "winners", better = NULL)
```

Arguments

`.data`	The dataset containing the columns related to Environments, Genotypes, and the response variable(s).
`env`	The name of the column that contains the levels of the environments.
`gen`	The name of the column that contains the levels of the genotypes.
`resp`	The response variable(s). To analyze multiple variables in a single procedure a vector of variables may be used. For example `resp = c(var1,var2,var3)`. Select helpers are also allowed.
`type`	The type of results. Defaults to `"winners"` (default), i.e., a two-way table with the winner genotype in each environment. If `type = "ranks"` return the genotype ranking within each environment.
`better`	A vector of the same length of the number of variables to rank the genotypes according to the response variable. Each element of the vector must be one of the `'h'` or `'l'`. If `'h'` is used (default), the genotypes are ranked from maximum to minimum. If `'l'` is used then the are ranked from minimum to maximum. Use a comma-separated vector of names. For example, `better = c("h,h,h,h,l")`, for ranking the fifth variable from minimum to maximum.

Value

A tibble with two-way table with the winner genotype in each environment (default) or the genotype ranking for each environment (if `type = "ranks"`).

Author(s)

Tiago Olivoto <tiagoolivoto@gmail.com>

Examples

```
library(metan)
ge_winners(data_ge, ENV, GEN, resp = everything())

# Assuming that for 'GY' lower values are better.
ge_winners(data_ge, ENV, GEN,
           resp = everything(),
           better = c("l, h"))

# Show the genotype ranking for each environment
ge_winners(data_ge, ENV, GEN,
           resp = everything(),
           type = "ranks")
```

---

gge *Genotype plus genotype-by-environment model*

Description

Produces genotype plus genotype-by-environment model based on a multi-environment trial dataset containing at least the columns for genotypes, environments and one response variable or a two-way table.

Usage

```
gge(
  .data,
  env,
  gen,
  resp,
  centering = "environment",
  scaling = "none",
  svp = "environment"
)
```

Arguments

`.data`	The dataset containing the columns related to Environments, Genotypes and the response variable(s). It is also possible to use a two-way table with genotypes in lines and environments in columns as input. In this case you must use `table = TRUE`.
`env`	The name of the column that contains the levels of the environments.
`gen`	The name of the column that contains the levels of the genotypes.
`resp`	The response variable(s). To analyze multiple variables in a single procedure a vector of variables may be used. For example `resp = c(var1,var2,var3)`. Select helpers are also supported.
`centering`	The centering method. Must be one of the `'none | 0'`, for no centering; `'global | 1'`, for global centered (E+G+GE); `'environment | 2'` (default), for environment-centered (G+GE); or `'double | 3'`, for double centred (GE). A biplot cannot be produced with models produced without centering.
`scaling`	The scaling method. Must be one of the `'none | 0'` (default), for no scaling; or `'sd | 1'`, where each value is divided by the standard deviation of its corresponding environment (column). This will put all environments roughly he same rang of values.
`svp`	The method for singular value partitioning. Must be one of the `'genotype | 1'`, (The singular value is entirely partitioned into the genotype eigenvectors, also called row metric preserving); `'environment | 2'`, default, (The singular value is entirely partitioned into the environment eigenvectors, also called column metric preserving); or `'symmetrical | 3'` (The singular value is symmetrically partitioned into the genotype and the environment eigenvectors This SVP is most often used in AMMI analysis and other biplot analysis, but it is not ideal for visualizing either the relationship among genotypes or that among the environments).

Value

The function returns a list of class gge containing the following objects

- **coordgen** The coordinates for genotypes for all components.
- **coordenv** The coordinates for environments for all components.
- **eigenvalues** The vector of eigenvalues.
- **totalvar** The overall variance.
- **labelgen** The name of the genotypes.
- **labelenv** The names of the environments.
- **labelaxes** The axes labels.
- **ge_mat** The data used to produce the model (scaled and centered).
- **centering** The centering method.
- **scaling** The scaling method.
- **svp** The singular value partitioning method.

- **d** The factor used to generate in which the ranges of genotypes and environments are comparable when singular value partitioning is set to 'genotype' or 'environment'.
- **grand_mean** The grand mean of the trial.
- **mean_gen** A vector with the means of the genotypes.
- **mean_env** A vector with the means of the environments.
- **scale_var** The scaling vector when the scaling method is `'sd'`.

Author(s)

Tiago Olivoto <tiagoolivoto@gmail.com>

References

Yan, W., and M.S. Kang. 2003. GGE biplot analysis: a graphical tool for breeders, geneticists, and agronomists. CRC Press.

Examples

```
library(metan)
mod <- gge(data_ge, ENV, GEN, GY)
plot(mod)

# GGE model for all numeric variables
mod2 <- gge(data_ge2, ENV, GEN, resp = everything())
plot(mod2)

# If we have a two-way table with the mean values for
# genotypes and environments

table <- make_mat(data_ge, GEN, ENV, GY)
table
make_long(table) %>%
gge(ENV, GEN, Y) %>%
plot()
```

---

gm_mean *Geometric mean*

Description

Helper function to compute the geometric mean. The geometric mean is the *n*th root of *n* products.

Usage

```
gm_mean(x, ..., na.rm = TRUE)
```

Arguments

x	A numeric vector or a data frame.
...	Variables to compute the geometric mean. If no variable is informed and x is a data frame, all the numeric variables will be used.
na.rm	A logical value indicating whether NA values should be stripped before computation proceeds.

Value

The geometric mean(s) of x. If x is a numeric vector, the function returns a numeric value. If a data frame is used then a numeric vector with the geometric mean for each variable is returned.

Note

Not useful if there are elements with values <=0.

See Also

hm_mean

Examples

```
num <- c(1:10, 50)
gm_mean(num)

num_df <- make_mat(data_ge, ENV, GEN, GY)
gm_mean(num_df)
```

---

hm_mean *Harmonic mean*

Description

Helper function to compute the harmonic mean. The harmonic mean is the reciprocal of the arithmetic mean of the reciprocals.

Usage

```
hm_mean(x, ..., na.rm = TRUE)
```

Arguments

x	A numeric vector or a data frame.
...	Variables to compute the harmonic mean. If no variable is informed and x is a data frame, all the numeric variables will be used.
na.rm	A logical value indicating whether NA values should be stripped before computation proceeds.

Value

The harmonic mean(s) of `x`. If `x` is a numeric vector, the function returns a numeric value. If a data frame is used then a numeric vector with the harmonic mean for each variable is returned.

Note

Not useful if there are elements with values `<=0`.

See Also

`gm_mean`

Examples

```
num <- c(1:10, 50)
hm_mean(num)

num_df <- make_mat(data_ge, ENV, GEN, GY)
hm_mean(num_df)
```

Huehn *Huehn's stability statistics*

Description

Performs a stability analysis based on Huehn (1979) statistics. The four nonparametric measures of phenotypic stability are: S1 (mean of the absolute rank differences of a genotype over the n environments), S2 (variance among the ranks over the k environments), S3 (sum of the absolute deviations), and S6 (relative sum of squares of rank for each genotype).

Usage

```
Huehn(.data, env, gen, rep, resp, verbose = TRUE)
```

Arguments

`.data`	The dataset containing the columns related to Environments, Genotypes, replication/block and response variable(s).
`env`	The name of the column that contains the levels of the environments.
`gen`	The name of the column that contains the levels of the genotypes.
`rep`	The name of the column that contains the levels of the replications/blocks.
`resp`	The response variable(s). To analyze multiple variables in a single procedure use, for example, `resp = c(var1,var2,var3)`.
`verbose`	Logical argument. If `verbose = FALSE` the code will run silently.

Value

An object of class Huehn, which is a list containing the results for each variable used in the argument resp. For each variable, a tibble with the following columns is returned.

- **GEN** The genotype's code.
- **Y** The mean for the response variable.
- **S1** Mean of the absolute rank differences of a genotype over the n environments.
- **S2** variance among the ranks over the k environments.
- **S3** Sum of the absolute deviations.
- **S6** Relative sum of squares of rank for each genotype.

Author(s)

Tiago Olivoto <tiagoolivoto@gmail.com>

References

Huehn, V.M. 1979. Beitrage zur erfassung der phanotypischen stabilitat. EDV Med. Biol. 10:112.

Examples

```
library(metan)
out <- Huehn(data_ge2, ENV, GEN, REP, PH)
```

---

inspect *Check for common erros in multi-environment trial data*

Description

inspect() scans a data.frame object for errors that may affect the use of functions in metan. By default, all variables are checked regarding the class (numeric or factor), missing values, and presence of possible outliers. The function will return a warning if the data looks like unbalanced, has missing values or possible outliers.

Usage

```
inspect(.data, ..., plot = FALSE, threshold = 15, verbose = TRUE)
```

Arguments

.data	The data to be analyzed
...	The variables in .data to check. If no variable is informed, all the variables in .data are used.
plot	Create a plot to show the check? Defaults to FALSE.
threshold	Maximum number of levels allowed in a character / factor column to produce a plot. Defaults to 15.
verbose	Logical argument. If TRUE (default) then the results for checks are shown in the console.

Value

A tibble with the following variables:

- **Variable** The name of variable
- **Class** The class of the variable
- **Missing** Contains missing values?
- **Levels** The number of levels of a factor variable
- **Valid_n** Number of valid n (omit NAs)
- **Outlier** Contains possible outliers?

Author(s)

Tiago Olivoto <tiagoolivoto@gmail.com>

Examples

```
library(metan)
inspect(data_ge)

# Create a toy example with messy data
df <- data_ge2[-c(2, 30, 45, 134), c(1:5)]
df[c(1, 20, 50), c(4, 5)] <- NA
df[40, 5] <- df[40, 5] * 2

inspect(df, plot = TRUE)
```

int.effects *Data for examples*

Description

Data for examples

Author(s)

Tiago Olivoto <tiagoolivoto@gmail.com>

is.lpcor *Coerce to an object of class lpcor*

Description

Functions to check if an object is of class `lpcor`

Usage

```
is.lpcor(x)
```

Arguments

`x`	An object to check.

Value

A logical value `TRUE` or `FALSE`.

Examples

```
library(metan)
library(dplyr)
mt_num = mtcars %>% select_if(., is.numeric)
lpdata = as.lpcor(cor(mt_num[1:5]),
                  cor(mt_num[1:5]),
                  cor(mt_num[2:6]),
                  cor(mt_num[4:8]))
is.lpcor(lpdata)
```

is.split_factors *Check if an object is of class split_factors*

Description

Functions to check if an object is of class `split_factors`

Usage

```
is.split_factors(x)
```

Arguments

`x`	The input data.

Details

A dataframe may be easily coerced to be split into named subsets based on each combination of factors existing in the original dataframe. For example, if the original data has two columns, namely ENV (four levels) and HIB (ten levels), and ten numeric columns, then using `as.split_factors` will split the data into 40 10-columns subsets, corresponding to each combination of ENV x HIB.

Value

A logical value `TRUE` or `FALSE`.

Examples

```
library(metan)
spdata = as.split_factors(iris)
is.split_factors(spdata)
```

lpcor *Linear and Partial Correlation Coefficients*

Description

Estimates the linear and partial correlation coefficients using as input a data frame or a correlation matrix.

Usage

```
lpcor(.data, ..., by = NULL, n = NULL, method = "pearson", verbose = TRUE)
```

Arguments

.data	The data to be analyzed. Must be a symmetric correlation matrix or, a dataframe containing the predictor variables, or an object of class split_factors.
...	Variables to use in the correlation. If ... is null (Default) then all the numeric variables from .data are used. It must be a single variable name or a comma-separated list of unquoted variables names.
by	One variable (factor) to split the data into subsets. The function is then applied to each subset and returns a list where each element contains the results for one level of the variable in by. To split the data by more than one factor variable, use the function split_factors to pass subsetted data to .data.
n	If a correlation matrix is provided, then n is the number of objects used to compute the correlation coefficients.
method	a character string indicating which correlation coefficient is to be computed. One of 'pearson' (default), 'kendall', or 'spearman'.
verbose	If verbose = TRUE then some results are shown in the console.

Value

If a grouping factor is used then a list is returned with the following values.

- **linear.mat** The matrix of linear correlation.
- **partial.mat** The matrix of partial correlations.
- **results** Hypothesis testing for each pairwise comparison.

Author(s)

Tiago Olivoto <tiagoolivoto@gmail.com>

Examples

```
library(metan)
partial1 <- pcor(iris)

# Alternatively using the pipe operator %>%
partial2 <- iris %>% lpcor()

# Using a correlation matrix
partial3 <- cor(iris[1:4]) %>%
            lpcor(n = nrow(iris),
                  verbose = FALSE)

# Select all numeric variables and compute the partial correlation
# For each level of \code{Species}

partial4 <- lpcor(iris, everithig(), by = Species)
print(partial4$summary)
```

mahala *Mahalanobis Distance*

Description

Compute the Mahalanobis distance of all pairwise rows in `.means`. The result is a symmetric matrix containing the distances that may be used for hierarchical clustering.

Usage

```
mahala(.means, covar, inverted = FALSE)
```

Arguments

`.means`	A matrix of data with, say, p columns.
`covar`	The covariance matrix.
`inverted`	Logical argument. If `TRUE`, `covar` is supposed to contain the inverse of the covariance matrix.

Value

A symmetric matrix with the Mahalanobis' distance.

Author(s)

Tiago Olivoto <tiagoolivoto@gmail.com>

Examples

```
library(metan)
library(dplyr)
# Compute the mean for genotypes
means = data_ge %>%
        select(-c(ENV, REP)) %>%
        group_by(GEN) %>%
        summarise_all(mean) %>%
        select(-GEN)

# Compute the covariance matrix
covmat = cov(means)

# Compute the distance
dist = mahala(means, covmat)

# Dendrogram
dend = as.dendrogram(hclust(as.dist(dist)))
plot(dend)
```

mahala_design *Mahalanobis distance from designed experiments*

Description

Compute the Mahalanobis distance using data from an experiment conducted in a randomized complete block design or completely randomized design.

Usage

```
mahala_design(
  .data,
  gen,
  rep,
  resp,
  design = "RCBD",
  by = NULL,
  return = "distance"
)
```

Arguments

.data	The dataset containing the columns related to Genotypes, replication/block and response variables. Alternatively, it is possible to use an object of class 'split_factors' to compute the distance for each level of the grouping factor. See ?split_factors.
gen	The name of the column that contains the levels of the genotypes.
rep	The name of the column that contains the levels of the replications/blocks.
resp	The response variables. For example resp = c(var1,var2,var3).
design	The experimental design. Must be RCBD or CRD.
by	One variable (factor) to split the data into subsets. The function is then applied to each subset and returns a list where each element contains the results for one level of the variable in by. To split the data by more than one factor variable, use the function split_factors to pass subsetted data to .data.
return	What the function return? Default is 'distance', i.e., the Mahalanobis distance. Alternatively, it is possible to return the matrix of means return = 'means', or the variance-covariance matrix of residuals return = 'covmat'.

Value

A symmetric matrix with the Mahalanobis' distance. If the .data is an object of class split_factors then a list of distances will be returned.

Author(s)

Tiago Olivoto <tiagoolivoto@gmail.com>

Examples

```
library(metan)
maha_group <- mahala_design(data_ge,
                            gen = GEN,
                            rep = REP,
                            resp = c(GY, HM))

# Compute one distance for each environment
maha_group <- mahala_design(data_ge, GEN, REP, everything(), by = ENV)

# Return the variance-covariance matrix of residuals
cov_mat <- mahala_design(data_ge,
                         gen = GEN,
                         rep = REP,
                         resp = c(GY, HM),
                         return = 'covmat')
```

---

make_long *Two-way table to a 'long' format*

Description

Helps users to easily convert a two-way table (genotype vs environment) to a 'long' format data. The data in `mat` will be gathered into three columns. The row names will compose the first column. The column names will compose the second column and the third column will contain the data that fills the two-way table.

Usage

```
make_long(mat, gen_in = "rows")
```

Arguments

`mat`	A two-way table. It must be a matrix or a data.frame with rownames.
`gen_in`	Where are the genotypes? Defaults to `'rows'`. If genotypes are in columns and environments in rows, set to `gen_in = 'cols'`.

Value

A tibble with three columns: GEN (genotype), ENV (environment), and Y (response) variable.

Author(s)

Tiago Olivoto <tiagoolivoto@gmail.com>

Examples

```
library(metan)

set.seed(1)
mat <- matrix(rnorm(9, 2530, 350), ncol = 3)
colnames(mat) <- paste("E", 1:3, sep = "")
rownames(mat) <- paste("G", 1:3, sep = "")

make_long(mat)

gen_cols <- t(mat)
make_long(gen_cols, gen_in = "cols")
```

`make_lower_tri` *Make a lower triangular matrix*

Description

This function help users to easily make a lower triangular matrix using a symmetric matrix.

Usage

```
make_lower_tri(x, diag = NA)
```

Arguments

`x`	A symmetric matrix
`diag`	What show in the diagonal of the matrix. Default to `NA`.

Value

A lower triangular matrix

Author(s)

Tiago Olivoto <tiagoolivoto@gmail.com>

Examples

```
library(metan)
m <- cor(data_ge2[,5:10])
make_upper_tri(m)
```

make_mat *Make a two-way table*

Description

This function help users to easily make a two-way table from a "long format" data.

Usage

```
make_mat(.data, row, col, value, fun = mean)
```

Arguments

.data	The dataset. Must contains at least two categorical columns.
row	The column of data in which the mean of each level will correspond to **one line** in the output.
col	The column of data in which the mean of each level will correspond to **one column** in the output.
value	The column of data that contains the values to fill the two-way table.
fun	The function to apply. Defaults to mean, i.e., the two-way table will show the mean values for each genotype-environment combination. Other R base functions such as max, min, sd, var, or an own function that return a single numeric value can be used.

Value

A two-way table with the argument row in the rows, col in the columns, filled by the argument value.

Author(s)

Tiago Olivoto <tiagoolivoto@gmail.com>

Examples

```
library(metan)
matrix = data_ge %>% make_mat(row = GEN, col = ENV, val = GY)
matrix

# An own function (sem, standart error of mean)

sem = function(data){
return(sd(data) / sqrt(length(data)))
}

data_ge %>% make_mat(GEN, ENV, GY, sem)
```

make_sym *Make a symmetric matrix on a triangular matrix*

Description

This function help users to easily make a symmetric matrix using a lower or an upper triangular matrix.

Usage

```
make_sym(.matrix, make = "upper", diag = NA)
```

Arguments

.matrix	The upper or lower triangular matrix.
make	The triangular to built. Default is "upper". In this case, a symmetric matrix will be built based on the values of a lower triangular matrix.
diag	What show in the diagonal of the matrix. Default to NA.

Value

A symmetric matrix.

Author(s)

Tiago Olivoto <tiagoolivoto@gmail.com>

Examples

```
library(metan)
lower_tri <- make_lower_tri(matrix(20, 4, 4))
lower_tri
make_sym(lower_tri)

upper_tri <- make_upper_tri(matrix(20, 4, 4))
upper_tri
make_sym(upper_tri, make = "lower", diag = 1)
```

make_upper_tri *Make an upper triangular matrix*

Description

This function help users to easily make an upper triangular matrix using a symmetric matrix.

Usage

```
make_upper_tri(x, diag = NA)
```

Arguments

x	A symmetric matrix
diag	What show in the diagonal of the matrix. Default to NA.

Value

An upper triangular matrix

Author(s)

Tiago Olivoto <tiagoolivoto@gmail.com>

Examples

```
library(metan)
m <- cor(data_ge2[,5:10])
make_upper_tri(m)
```

meansGxE *Data for examples*

Description

This dataset contains the means for grain yield of 10 genotypes cultivated in 5 environments. The interaction effects for this data is found in int.effects

Author(s)

Tiago Olivoto <tiagoolivoto@gmail.com>

means_by *Means by one or more factors*

Description

Computes the mean for all numeric variables of a data frame, grouping by one or more factors.

Usage

```
means_by(.data, ...)
```

Arguments

.data	A data frame
...	One or more categorical variables for grouping the data.

Value

An object of class tbl_df with the computed means by each level of the factor(s) declared in ....

Examples

```
library(metan)
means_by(data_ge2, ENV)
means_by(data_ge2, GEN, ENV)
```

mtsi *Multi-trait stability index*

Description

Computes the multi-trait stability index proposed by Olivoto et al. (2019)

Usage

```
mtsi(.data, index = "waasby", SI = 15, mineval = 1, verbose = TRUE)
```

Arguments

.data	An object of class waasb or waas.
index	If index = 'waasby' (default) both stability and mean performance are considered. If index = 'waasb' the multi-trait index will be computed considering the stability of genotypes only. More details can be seen in waasb and waas functions.
SI	An integer (0-100). The selection intensity in percentage of the total number of genotypes.
mineval	The minimum value so that an eigenvector is retained in the factor analysis.
verbose	If verbose = TRUE (Default) then some results are shown in the console.

Value

An object of class mtsi with the following items:

- **data** The data used to compute the factor analysis.
- **cormat** The correlation matrix among the environments.
- **PCA** The eigenvalues and explained variance.
- **FA** The factor analysis.
- **KMO** The result for the Kaiser-Meyer-Olkin test.
- **MSA** The measure of sampling adequacy for individual variable.
- **communalities** The communalities.
- **communalities.mean** The communalities' mean.
- **initial.loadings** The initial loadings.
- **finish.loadings** The final loadings after varimax rotation.
- **canonical.loadings** The canonical loadings.
- **scores.gen** The scores for genotypes in all retained factors.
- **scores.ide** The scores for the ideotype in all retained factors.
- **MTSI** The multi-trait stability index.
- **contri.fac** The relative contribution of each factor on the MTSI value. The lower the contribution of a factor, the close of the ideotype the variables in such factor are.
- **sel.dif** The selection differential for the WAASBY or WAASB index.
- **mean.sd** The mean for the differential selection.
- **sel.dif.var** The selection differential for the variables.
- **Selected** The selected genotypes.

Author(s)

Tiago Olivoto <tiagoolivoto@gmail.com>

References

Olivoto, T., A.D.C. L\'ucio, J.A.G. da silva, B.G. Sari, and M.I. Diel. 2019. Mean performance and stability in multi-environment trials II: Selection based on multiple traits. Agron. J. (in press).

Examples

```
library(metan)

# Based on stability only, for both GY and HM, higher is better
mtsi_model <- waasb(data_ge,
                    env = ENV,
                    gen = GEN,
                    rep = REP,
                    resp = c(GY, HM))
mtsi_index <- mtsi(mtsi_model, index = 'waasb')


# Based on mean performance and stability (using pipe operator %>%)
# GY: higher is better
# HM: lower is better

mtsi_index2 <- data_ge %>%
 waasb(ENV, GEN, REP,
       resp = c(GY, HM),
       mresp = c(100, 0)) %>%
 mtsi()
```

`pairs_mantel` *Mantel test for a set of correlation matrices*

Description

This function generate a pairwise matrix of plots to compare the similarity of two or more correlation matrices. In the upper diagonal are presented the plots and in the lower diagonal the result of Mantel test based on permutations.

Usage

```
pairs_mantel(
  ...,
  type = 1,
  nrepet = 1000,
  names = NULL,
  prob = 0.05,
  diag = FALSE,
  export = FALSE,
  main = "auto",
  file.type = "pdf",
  file.name = NULL,
  width = 8,
  height = 7,
```

```
  resolution = 300,
  size.point = 0.5,
  shape.point = 19,
  alpha.point = 1,
  fill.point = NULL,
  col.point = "black",
  minsize = 2,
  maxsize = 3,
  signcol = "green",
  alpha = 0.15,
  diagcol = "gray",
  col.up.panel = "gray",
  col.lw.panel = "gray",
  col.dia.panel = "gray",
  pan.spacing = 0.15,
  digits = 2
)
```

Arguments

`...`	The input matrices. May be an output generated by the function `lpcor` or a coerced list generated by the function `as.lpcor`
`type`	The type of correlation if an obect generated by the function `lpcor` is used. 1 = Linear correlation matrices, or 2 = partial correlation matrices.
`nrepet`	The number of permutations. Default is 1000
`names`	An optional vector of names of the same length of `...` .
`prob`	The error probability for Mantel test.
`diag`	Logical argument. If `TRUE`, the Kernel density is shown in the diagonal of plot.
`export`	Logical argument. If `TRUE`, then the plot is exported to the current directory.
`main`	The title of the plot, set to 'auto'.
`file.type`	The format of the file if `export = TRUE`. Set to `'pdf'`. Other possible values are `*.tiff` using `file.type = 'tiff'`.
`file.name`	The name of the plot when exported. Set to `NULL`, i.e., automatically.
`width`	The width of the plot, set to `8`.
`height`	The height of the plot, set to `7`.
`resolution`	The resolution of the plot if `file.type = 'tiff'` is used. Set to `300` (300 dpi).
`size.point`	The size of the points in the plot. Set to `0.5`.
`shape.point`	The shape of the point, set to `19`.
`alpha.point`	The value for transparency of the points: 1 = full color.
`fill.point`	The color to fill the points. Valid argument if points are between 21 and 25.
`col.point`	The color for the edge of the point, set to `black`.
`minsize`	The size of the letter that will represent the smallest correlation coefficient.
`maxsize`	The size of the letter that will represent the largest correlation coefficient.

signcol	The colour that indicate significant correlations (based on the prob value.), set to 'green'.
alpha	The value for transparency of the color informed in signcol, when 1 = full color. Set to 0.15.
diagcol	The color in the kernel distribution. Set to 'gray'.
col.up.panel, col.lw.panel, col.dia.panel	The color for the opper, lower and diagonal pannels. Set to 'gray', 'gray', and 'gray', respectively.
pan.spacing	The space between the pannels. Set to 0.15.
digits	The number of digits to show in the plot.

Value

An object of class gg,ggmatrix.

Author(s)

Tiago Olivoto <tiagoolivoto@gmail.com>

Examples

```
library(metan)
library(dplyr)
# iris dataset
lpc <- iris %>%
       split_factors(Species) %>%
       lpcor() %>%
       pairs_mantel(names = c('setosa', 'versicolor', 'virginica'))


# mtcars dataset
mt_num <- mtcars %>% select_if(., is.numeric)
lpdata <- as.lpcor(cor(mt_num[1:5]),
                   cor(mt_num[1:5]),
                   cor(mt_num[2:6]),
                   cor(mt_num[4:8])) %>%
          pairs_mantel()
```

---

path_coeff *Path coefficients with minimal multicollinearity*

Description

Computes direct and indirect effects in path analysis. An algorithm to select a set of predictors with minimal multicollinearity and high explanatory power is implemented.

Usage

```
path_coeff(
  .data,
  resp,
  by = NULL,
  pred = NULL,
  exclude = FALSE,
  correction = NULL,
  knumber = 50,
  brutstep = FALSE,
  maxvif = 10,
  missingval = "pairwise.complete.obs",
  verbose = TRUE
)
```

Arguments

.data	The data. Must be a dataframe or an object of class split_factors.
resp	The dependent variable.
by	One variable (factor) to split the data into subsets. The function is then applied to each subset and returns a list where each element contains the results for one level of the variable in by. To split the data by more than one factor variable, use the function split_factors to pass subsetted data to .data.
pred	The predictor variables, set to NULL, i.e., the predictor variables are all the numeric variables in the data except that in resp.
exclude	Logical argument, set to false. If exclude = TRUE, then the variables in pred are deleted from the data, and the analysis will use as predictor those that remained, except that in resp.
correction	Set to NULL. A correction value (k) that will be added into the diagonal elements of the **X'X** matrix aiming at reducing the harmful problems of the multicollinearity in path analysis (Olivoto et al., 2017)
knumber	When correction = NULL, a plot showing the values of direct effects in a set of different k values (0-1) is produced. knumber is the number of k values used in the range of 0 to 1.
brutstep	Logical argument, set to FALSE. If true, then an algorithm will select a subset of variables with minimal multicollinearity and fit a set of possible models. See the **Details** section for more information.
maxvif	The maximum value for the Variance Inflation Factor (cut point) that will be accepted. See the **Details** section for more information.
missingval	How to deal with missing values. For more information, please see ?cor.
verbose	If verbose = TRUE then some results are shown in the console.

Details

When brutstep = TRUE, first, the algorithm will select a set of predictors with minimal multicollinearity. The selection is based on the variance inflation factor (VIF). An iterative process is

performed until the maximum VIF observed is less than `maxvif`. The variables selected in this iterative process are then used in a series of stepwise-based regressions. The first model is fitted and p-1 predictor variables are retained (p is the number of variables selected in the iterative process. The second model adjusts a regression considering p-2 selected variables, and so on until the last model, which considers only two variables. Three objects are created. `Summary`, with the process summary, `Models`, containing the aforementioned values for all the adjusted models; and `Selectedpred`, a vector with the name of the selected variables in the iterative process.

Value

An object of class `path_coeff`, `group_path`, or `brute_path` with the following items:

- **Corr.x** A correlation matrix between the predictor variables.
- **Corr.y** A vector of correlations between each predictor variable with the dependent variable.
- **Coefficients** The path coefficients. Direct effects are the diagonal elements, and the indirect effects those in the off-diagonal elements (column)
- **Eigen** Eigenvectors and eigenvalues of the `Corr.x`.
- **VIF** The Variance Inflation Factors.
- **plot** A ggplot2-based graphic showing the direct effects in 21 different k values.
- **Predictors** The predictor variables used in the model.
- **CN** The Condition Number, i.e., the ratio between the highest and lowest eigenvalue.
- **Det** The matrix determinant of the `Corr.x`..
- **R2** The coefficient of determination of the model.
- **Residual** The residual effect of the model.
- **Response** The response variable.
- **weightvar** The order of the predictor variables with the highest weight (highest eigenvector) in the lowest eigenvalue.

If `.data` is an object of class `split_factors` then a list is returned with the above items for each level of the grouping variable in the function `split_factors`.

Author(s)

Tiago Olivoto <tiagoolivoto@gmail.com>

References

Olivoto, T., V.Q. Souza, M. Nardino, I.R. Carvalho, M. Ferrari, A.J. Pelegrin, V.J. Szareski, and D. Schmidt. 2017. Multicollinearity in path analysis: a simple method to reduce its effects. Agron. J. 109:131-142. doi:10.2134/agronj2016.04.0196. 10.2134/agronj2016.04.0196.

Examples

```
library(metan)

# Using KW as the response variable and all other ones as predictors
pcoeff <- path_coeff(data_ge2, resp = KW)


# Declaring the predictors
pcoeff2 <- path_coeff(data_ge2,
                      resp = KW,
                      pred = c(PH, EH, NKE, TKW))


# Selecting variables to be excluded from the analysis
pcoeff3 <- path_coeff(data_ge2,
                    resp = KW,
                    pred = c(NKR, PERK, KW, NKE),
                    exclude = TRUE)


# Selecting a set of predictors with minimal multicollinearity
# Maximum variance Inflation factor of 5
pcoeff4 <- path_coeff(data_ge2,
                    resp = KW,
                    brutstep = TRUE,
                    maxvif = 5)


# When one analysis should be carried out for each environment
# Using the forward-pipe operator %>%
pcoeff5 <- path_coeff(data_ge2, resp = KW, by = ENV)
```

---

performs_ammi *Additive Main effects and Multiplicative Interaction*

Description

Compute the Additive Main effects and Multiplicative interaction. This function also serves as a helper function for other procedures performed in the **metan** package such as `waas` and `wsmp`

Usage

```
performs_ammi(.data, env, gen, rep, resp, verbose = TRUE)
```

Arguments

.data	The dataset containing the columns related to Environments, Genotypes, replication/block and response variable(s).
env	The name of the column that contains the levels of the environments
gen	The name of the column that contains the levels of the genotypes
rep	The name of the column that contains the levels of the replications/blocks
resp	The response variable(s). To analyze multiple variables in a single procedure a vector of variables may be used. For example resp = c(var1,var2,var3).
verbose	Logical argument. If verbose = FALSE the code will run silently.

Value

- **ANOVA** The analysis of variance for the AMMI model.
- **PCA** The principal component analysis
- **MeansGxE** The means of genotypes in the environments
- **model** scores for genotypes and environments in all the possible axes.

Author(s)

Tiago Olivoto <tiagoolivoto@gmail.com>

Examples

```
library(metan)
ammi_model = performs_ammi(data_ge, ENV, GEN, REP,
                           resp = c(GY, HM))

# GY x PC1 (variable GY)
plot_scores(ammi_model,
            col.env = 'olivedrab',
            col.gen = 'orange2',
            x.lab = 'My own x label')

# PC1 x PC2 (variable HM)
plot_scores(ammi_model,
            type = 2)

# PC1 x PC2 (variable HM)
# Draw a convex hull polygon
plot_scores(ammi_model,
            type = 2,
            polygon = TRUE)
```

plot.can_cor *Plots an object of class can_cor*

Description

Graphs of the Canonical Correlation Analysis

Usage

```
## S3 method for class 'can_cor'
plot(
  x,
  type = 1,
  plot_theme = theme_metan(),
  size.tex.lab = 12,
  size.tex.pa = 3.5,
  x.lab = NULL,
  x.lim = NULL,
  x.breaks = waiver(),
  y.lab = NULL,
  y.lim = NULL,
  y.breaks = waiver(),
  axis.expand = 1.1,
  shape = 21,
  col.shape = "orange",
  col.alpha = 0.9,
  size.shape = 3.5,
  size.bor.tick = 0.3,
  labels = FALSE,
  main = NULL,
  ...
)
```

Arguments

x	The waasb object
type	The type of the plot. Defaults to type = 1 (Scree-plot of the correlations of the canonical loadings). Use type = 2, to produce a plot with the scores of the variables in the first group, type = 3 to produce a plot with the scores of the variables in the second group, or type = 4 to produce a circle of correlations.
plot_theme	The graphical theme of the plot. Default is plot_theme = theme_metan(). For more details,see theme.
size.tex.lab	The size of the text in axis text and labels.
size.tex.pa	The size of the text of the plot area. Default is 3.5.
x.lab	The label of x-axis. Each plot has a default value. New arguments can be inserted as x.lab = 'my label'.

x.lim	The range of x-axis. Default is NULL (maximum and minimum values of the data set). New arguments can be inserted as x.lim = c(x.min,x.max).
x.breaks	The breaks to be plotted in the x-axis. Default is authomatic breaks. New arguments can be inserted as x.breaks = c(breaks)
y.lab	The label of y-axis. Each plot has a default value. New arguments can be inserted as y.lab = 'my label'.
y.lim	The range of y-axis. Default is NULL. The same arguments than x.lim can be used.
y.breaks	The breaks to be plotted in the x-axis. Default is authomatic breaks. The same arguments than x.breaks can be used.
axis.expand	Multiplication factor to expand the axis limits by to enable fitting of labels. Default is 1.1.
shape	The shape of points in the plot. Default is 21 (circle). Values must be between 21-25: 21 (circle), 22 (square), 23 (diamond), 24 (up triangle), and 25 (low triangle).
col.shape	A vector of length 2 that contains the color of shapes for genotypes above and below of the mean, respectively. Defaults to "orange". c("blue","red").
col.alpha	The alpha value for the color. Default is 0.9. Values must be between 0 (full transparency) to 1 (full color).
size.shape	The size of the shape in the plot. Default is 3.5.
size.bor.tick	The size of tick of shape. Default is 0.3. The size of the shape will be size.shape + size.bor.tick
labels	Logical arguments. If TRUE then the points in the plot will have labels.
main	The title of the plot. Defaults to NULL, in which each plot will have a default title. Use a string text to create an own title or set to main = FALSE to omit the plot title.
...	Currently not used.

Value

An object of class gg,ggplot.

Author(s)

Tiago Olivoto <tiagoolivoto@gmail.com>

Examples

```
library(metan)
cc1 = can_corr(data_ge2,
               FG = c(PH, EH, EP),
               SG = c(EL, ED, CL, CD, CW, KW, NR))
plot(cc1, 2)
cc2 = can_corr(data_ge2,
               FG = c(PH, EH, EP),
```

```
                SG = c(EL, ED, CL, CD, CW, KW, NR),
                means_by = GEN)
plot(cc2, 2, labels = TRUE)
```

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

Description

Plot an object of class clustering

Usage

```
## S3 method for class 'clustering'
plot(x, horiz = TRUE, type = "dendrogram", ...)
```

Arguments

`x`	An object of class `clustering`
`horiz`	Logical indicating if the dendrogram should be drawn horizontally or not.
`type`	The type of plot. Must be one of the 'dendrogram' or 'cophenetic'.
`...`	Other arguments passed from the function `plot.dendrogram` or `abline`.

Value

An object of class `gg,ggplot` if `type == "cophenetic"`.

Author(s)

Tiago Olivoto <tiagoolivoto@gmail.com>

Examples

```
# Define 4 clusters
d = clustering(data_ge2,
                means_by = GEN,
                scale = TRUE,
                nclust = 4)
plot(d)
```

plot.corr_coef *Create a correlation heat map*

Description

Create a correlation heat map for object of class corr_coef

Usage

```
## S3 method for class 'corr_coef'
plot(
  x,
  type = "lower",
  diag = FALSE,
  reorder = TRUE,
  digits = 2,
  col.low = "blue",
  col.mid = "white",
  col.high = "red",
  lab.x.position = NULL,
  lab.y.position = NULL,
  legend.position = NULL,
  legend.title = "Pearson's\nCorrelation",
  size.text.plot = 3,
  size.text.lab = 10,
  ...
)
```

Arguments

x	The data set.
type	The type of heat map to produce. Either lower (default) to produce a lower triangle heat map or upper to produce an upper triangular heat map.
diag	Plot diagonal elements? Defaults to FALSE.
reorder	Reorder the correlation matrix to identify the hidden pattern? Defaults to FALSE.
digits	The digits to show in the heat map.
col.low, col.mid, col.high	The color for the low (-1), mid(0) and high (1) points in the color key. Defaults to blue, white, and red, respectively.
lab.x.position, lab.y.position	The position of the x and y axis label. Defaults to "bottom" and "right" if type = "lower" or "top" and "left" if type = "upper".
legend.position	The legend position in the plot.
legend.title	The title of the color key. Defaults to "Pearson's Correlation".

size.text.plot, size.text.lab
: The size of the text in plot area (Defaults to 3) and labels (Defaults to 10), respectively. triangle heatmap.

...
: Not used currently.

Value

An object of class gg,ggplot

Author(s)

Tiago Olivoto <tiagoolivoto@gmail.com>

Examples

```
library(metan)
# All numeric variables
all <- corr_coef(data_ge2)
plot(all)
plot(all, reorder = FALSE)

# Select variables
sel <- corr_coef(data_ge2, EP, EL, CD, CL)
plot(sel,
     type = "upper",
     reorder = FALSE,
     size.text.lab = 14,
     size.text.plot = 5)
```

---

plot.cvalidation *Plot the RMSPD of a cross-validation procedure*

Description

Boxplot showing the Root Means Square Prediction Difference of of a cross validation procedure.

Usage

```
## S3 method for class 'cvalidation'
plot(
  x,
  violin = FALSE,
  export = FALSE,
  order_box = FALSE,
  x.lab = NULL,
  y.lab = NULL,
  size.tex.lab = 12,
  file.type = "pdf",
  file.name = NULL,
```

```
  plot_theme = theme_metan(),
  width = 6,
  height = 6,
  resolution = 300,
  col.violin = "gray90",
  col.boxplot = "gray70",
  col.boxplot.win = "cyan",
  width.boxplot = 0.6,
  x.lim = NULL,
  x.breaks = waiver(),
  ...
)
```

Arguments

x	An object of class cvalidation fitted with the functions cv_ammi, cv_ammif, cv_blup, or a bound object fitted with bind_cv.
violin	Define if a violin plot is used with boxplot. Default is 'TRUE'
export	Export (or not) the plot. Default is T.
order_box	Logical argument. If TRUE then the boxplots will be ordered according to the values of the RMSPD.
x.lab	The label of x-axis. New arguments can be inserted as x.lab = 'my x label'.
y.lab	The label of y-axis. New arguments can be inserted as y.lab = 'my y label'.
size.tex.lab	The size of the text in axis text and labels.
file.type	The type of file to be exported. Default is pdf, Graphic can also be exported in *.tiff format by declaring file.type = 'tiff'.
file.name	The name of the file for exportation, default is NULL, i.e. the files are automatically named.
plot_theme	The graphical theme of the plot. Default is plot_theme = theme_metan(). For more details,see theme.
width	The width 'inch' of the plot. Default is 6.
height	The height 'inch' of the plot. Default is 6.
resolution	The resolution of the plot. Parameter valid if file.type = 'tiff' is used. Default is 300 (300 dpi)
col.violin	Parameter valid if violin = T. Define the color of the violin plot. Default is 'gray90.
col.boxplot	Define the color for boxplot. Default is 'gray70'.
col.boxplot.win	Define the color for boxplot of the best model. Default is 'cyan'.
width.boxplot	The width of boxplots. Default is 0.2.
x.lim	The range of x-axis. Default is NULL (maximum and minimum values of the data set). New arguments can be inserted as x.lim = c(x.min,x.max).
x.breaks	The breaks to be plotted in the x-axis. Default is authomatic breaks. New arguments can be inserted as x.breaks = c(breaks)
...	Currently not used.

Details

Five statistics are shown in this type of plot. The lower and upper hinges correspond to the first and third quartiles (the 25th and 75th percentiles). The upper whisker extends from the hinge to the largest value no further than 1.5 * IQR from the hinge (where IQR is the inter-quartile range). The lower whisker extends from the hinge to the smallest value at most 1.5 * IQR of the hinge. Data beyond the end of the whiskers are considered outlying points.

Value

An object of class `gg,ggplot`.

Author(s)

Tiago Olivoto <tiagoolivoto@gmail.com>

Examples

```
validation <- cv_ammif(data_ge,
                       resp = GY,
                       gen = GEN,
                       env = ENV,
                       rep = REP,
                       nboot = 5)
plot(validation)
```

---

plot.env_dissimilarity

Plot an object of class env_dissimilarity

Description

Create dendrograms to show the dissimilarity between environments.

Usage

```
## S3 method for class 'env_dissimilarity'
plot(x, var = 1, nclust = NULL, ...)
```

Arguments

`x`	An object of class `env_dissimilarity`
`var`	The variable to plot. Defaults to `var = 1` the first variable of `x`.
`nclust`	The number of clusters to show.
`...`	Other arguments bo be passed to the function `hclust`.

Author(s)

Tiago Olivoto <tiagoolivoto@gmail.com>

Examples

```
library(metan)
mod <- env_dissimilarity(data_ge, ENV, GEN, REP, GY)
plot(mod)
```

---

plot.fai_blup *Multi-trait selection index*

Description

Plot the multitrait index based on factor analysis and ideotype-design proposed by Rocha et al. (2018).

Usage

```
## S3 method for class 'fai_blup'
plot(
  x,
  ideotype = 1,
  SI = 15,
  radar = TRUE,
  arrange.label = FALSE,
  size.point = 2,
  col.sel = "red",
  col.nonsel = "black",
  size.text = 10,
  ...
)
```

Arguments

x	An object of class waasb
ideotype	The ideotype to be plotted. Default is 1.
SI	An integer [0-100]. The selection intensity in percentage of the total number of genotypes.
radar	Logical argument. If true (default) a radar plot is generated after using coord_polar().
arrange.label	Logical argument. If TRUE, the labels are arranged to avoid text overlapping. This becomes useful when the number of genotypes is large, say, more than 30.
size.point	The size of the point in graphic.
col.sel	The colour for selected genotypes.

col.nonsel	The colour for nonselected genotypes.
size.text	The size for the text in the plot. Defaults to 10.
...	Other arguments to be passed from ggplot2::theme().

Value

An object of class gg,ggplot.

Author(s)

Tiago Olivoto <tiagoolivoto@gmail.com>

References

Rocha, J.R.A.S.C.R, J.C. Machado, and P.C.S. Carneiro. 2018. Multitrait index based on factor analysis and ideotype-design: proposal and application on elephant grass breeding for bioenergy. GCB Bioenergy 10:52-60. doi: doi:10.1111/gcbb.12443.

Examples

```
library(metan)

FAI = data_ge2 %>%
      waasb(ENV, GEN, REP, c(KW, NKE, PH, EH)) %>%
      fai_blup(DI = c('max, max, max, min'),
               UI = c('min, min, min, max'),
               SI = 15)
plot(FAI)
```

---

plot.ge_cluster *Plot an object of class ge_cluster*

Description

Plot an object of class ge_cluster

Usage

```
## S3 method for class 'ge_cluster'
plot(x, nclust = NULL, xlab = "", ...)
```

Arguments

x	An object of class ge_cluster
nclust	The number of clusters to show.
xlab	The label of the x axis.
...	Other arguments passed from the function plot.hclust.

Author(s)

Tiago Olivoto <tiagoolivoto@gmail.com>

---

plot.ge_effects *Plot an object of class ge_effects*

Description

Plot the regression model generated by the function ge_effects.

Usage

```
## S3 method for class 'ge_effects'
plot(
  x,
  var = 1,
  plot_theme = theme_metan(),
  x.lab = NULL,
  y.lab = NULL,
  leg.position = "right",
  size.text = 12,
  ...
)
```

Arguments

x	An object of class ge_effects
var	The variable to plot. Defaults to var = 1 the first variable of x.
plot_theme	The graphical theme of the plot. Default is plot_theme = theme_metan(). For more details, see theme.
x.lab	The label of x-axis. Each plot has a default value. New arguments can be inserted as x.lab = "my label".
y.lab	The label of y-axis. Each plot has a default value. New arguments can be inserted as y.lab = "my label".
leg.position	The position of the legend.
size.text	The size of the text in the axes text and labels. Default is 12.
...	Current not used.

Value

An object of class gg,ggplot.

Author(s)

Tiago Olivoto <tiagoolivoto@gmail.com>

See Also

ge_plot

Examples

```
library(metan)
ge_eff <- ge_effects(data_ge2, ENV, GEN, REP, PH)
plot(ge_eff)
```

---

plot.ge_factanal *Plot the ge_factanal model*

Description

This function plot the scores for genotypes obtained in the factor analysis to interpret the stability

Usage

```
## S3 method for class 'ge_factanal'
plot(
  x,
  var = 1,
  plot_theme = theme_metan(),
  x.lim = NULL,
  x.breaks = waiver(),
  x.lab = NULL,
  y.lim = NULL,
  y.breaks = waiver(),
  y.lab = NULL,
  shape = 21,
  col.shape = "gray30",
  col.alpha = 1,
  size.shape = 2.2,
  size.bor.tick = 0.3,
  size.tex.lab = 12,
  size.tex.pa = 3.5,
  force.repel = 1,
  line.type = "dashed",
```

```
  line.alpha = 1,
  col.line = "black",
  size.line = 0.5,
  ...
)
```

Arguments

x	An object of class ge_factanal
var	The variable to plot. Defaults to var = 1 the first variable of x.
plot_theme	The graphical theme of the plot. Default is plot_theme = theme_metan(). For more details, see theme.
x.lim	The range of x-axis. Default is NULL (maximum and minimum values of the data set). New arguments can be inserted as x.lim = c(x.min,x.max).
x.breaks	The breaks to be plotted in the x-axis. Default is authomatic breaks. New arguments can be inserted as x.breaks = c(breaks)
x.lab	The label of x-axis. Each plot has a default value. New arguments can be inserted as x.lab = "my label".
y.lim	The range of x-axis. Default is NULL. The same arguments than x.lim can be used.
y.breaks	The breaks to be plotted in the x-axis. Default is authomatic breaks. The same arguments than x.breaks can be used.
y.lab	The label of y-axis. Each plot has a default value. New arguments can be inserted as y.lab = "my label".
shape	The shape for genotype indication in the plot. Default is 1 (circle). Values between 21-25: 21 (circle), 22 (square), 23 (diamond), 24 (up triangle), and 25 (low triangle) allows a color for fill the shape.
col.shape	The shape color for genotypes. Must be one value or a vector of colors with the same length of the number of genotypes. Default is "gray30". Other values can be attributed. For example, transparent_color(), will make a plot with only an outline around the shape area.
col.alpha	The alpha value for the color. Default is 1. Values must be between 0 (full transparency) to 1 (full color).
size.shape	The size of the shape (both for genotypes and environments). Default is 2.2.
size.bor.tick	The size of tick of shape. Default is 0.3. The size of the shape will be size.shape + size.bor.tick
size.tex.lab	The size of the text in the axes text and labels. Default is 12.
size.tex.pa	The size of the text of the plot area. Default is 3.5.
force.repel	Force of repulsion between overlapping text labels. Defaults to 1.
line.type	The type of the line that indicate the means in the biplot. Default is "solid". Other values that can be attributed are: "blank", no lines in the biplot, "dashed","dotted","dotdash", "twodash".

line.alpha	The alpha value that combine the line with the background to create the appearance of partial or full transparency. Default is 0.4. Values must be between "0" (full transparency) to "1" (full color).
col.line	The color of the line that indicate the means in the biplot. Default is "gray"
size.line	The size of the line that indicate the means in the biplot. Default is 0.5.
...	Currently not used..

Value

An object of class gg,ggplot.

Author(s)

Tiago Olivoto <tiagoolivoto@gmail.com>

See Also

ge_factanal

Examples

```
library(metan)
library(ggplot2)
model = ge_factanal(data_ge2,
                    env = ENV,
                    gen = GEN,
                    rep = REP,
                    resp = PH)
plot(model)

plot(model,
     size.shape = 3,
     force.repel = 10,
     col.shape = "orange",
     col.line = "red")
```

---

plot.ge_reg *Plot an object of class ge_reg*

Description

Plot the regression model generated by the function ge_reg.

Usage

```
## S3 method for class 'ge_reg'
plot(
  x,
  var = 1,
  type = 1,
  plot_theme = theme_metan(),
  x.lim = NULL,
  x.breaks = waiver(),
  x.lab = NULL,
  y.lim = NULL,
  y.breaks = waiver(),
  y.lab = NULL,
  leg.position = "right",
  size.tex.lab = 12,
  ...
)
```

Arguments

x	An object of class ge_factanal
var	The variable to plot. Defaults to var = 1 the first variable of x.
type	The type of plot to show. type = 1 produces a plot with the environmental index in the x axis and the genotype mean yield in the y axis. type = 2 produces a plot with the response variable in the x axis and the slope of the regression in the y axis.
plot_theme	The graphical theme of the plot. Default is plot_theme = theme_metan(). For more details, see theme.
x.lim	The range of x-axis. Default is NULL (maximum and minimum values of the data set). New arguments can be inserted as x.lim = c(x.min,x.max).
x.breaks	The breaks to be plotted in the x-axis. Default is authomatic breaks. New arguments can be inserted as x.breaks = c(breaks)
x.lab	The label of x-axis. Each plot has a default value. New arguments can be inserted as x.lab = "my label".
y.lim	The range of x-axis. Default is NULL. The same arguments than x.lim can be used.
y.breaks	The breaks to be plotted in the x-axis. Default is authomatic breaks. The same arguments than x.breaks can be used.
y.lab	The label of y-axis. Each plot has a default value. New arguments can be inserted as y.lab = "my label".
leg.position	The position of the legend.
size.tex.lab	The size of the text in the axes text and labels. Default is 12.
...	Currently not used..

Value

An object of class gg,ggplot.

Author(s)

Tiago Olivoto <tiagoolivoto@gmail.com>

See Also

ge_factanal

Examples

```
library(metan)
model <- ge_reg(data_ge2, ENV, GEN, REP, PH)
plot(model)
```

---

plot.gge *Create GGE biplots*

Description

Produces a ggplot2-based GGE biplot based on a model of class gge. Since the output is an object of class ggplot, all stylistic attributes of the output can be customized using the power of plot customization provided by ggplot2.

Usage

```
## S3 method for class 'gge'
plot(
  x,
  var = 1,
  type = 1,
  sel_env = NA,
  sel_gen = NA,
  sel_gen1 = NA,
  sel_gen2 = NA,
  shape.gen = 21,
  shape.env = 23,
  size.shape = 2.2,
  size.shape.win = 3.2,
  size.bor.tick = 0.3,
  col.gen = "blue",
  col.env = "forestgreen",
  col.alpha = 1,
  col.circle = "gray",
```

```
  col.alpha.circle = 0.5,
  leg.lab = c("Gen", "Env"),
  size.text.gen = 4,
  size.text.env = 4,
  size.text.lab = 12,
  size.line = 0.8,
  large_label = 4.5,
  axis_expand = 1.2,
  title = TRUE,
  plot_theme = theme_metan(),
  ...
)
```

Arguments

x	An object of class gge
var	The variable to plot. Defaults to var = 1 the first variable of x.
type	The type of biplot to produce. 1. Basic biplot. 2. Mean performance vs. stability. 3. Which-won-where. 4. Discriminativeness vs. representativeness. 5. Examine an environment. 6. Ranking environments. 7. Examine a genotype. 8. Ranking genotypes. 9. Compare two genotypes. 10. Relationship among environments
sel_env, sel_gen	The name of the environment and genotype to examine when type = 5 and type = 7, respectively. Must be a string which matches a environment or genotype label.
sel_gen1, sel_gen2	The name of genotypes to compare between when type = 9. Must be a string present in the genotype's name.
shape.gen, shape.env	The shape for genotype and environment indication in the biplot. Defaults to shape.gen = 21 (circle) for genotypes and shape.env = 23 (rhombus) for environments. Values must be between 21-25: 21 (circle), 22 (square), 23 (rhombus), 24 (up triangle), and 25 (low triangle).
size.shape	The size of the shape (both for genotypes and environments). Defaults to 2.2.
size.shape.win	The size of the shape for winners genotypes when type = 3. Defaults to 3.2.
size.bor.tick	The size of tick of shape. Default is 0.3. The size of the shape will be size.shape + size.bor.tick

`col.gen, col.env`
: Color for genotype and environment attributes in the biplot. Defaults to `col.gen = 'blue'` and `col.env = 'forestgreen'`

`col.alpha`
: The alpha value for the color. Defaults to `1`. Values must be between `0` (full transparency) to 1 (full color).

`col.circle, col.alpha.circle`
: The color and alpha values for the circle lines. Defaults to `'gray'` and `0.4`, respectively.

`leg.lab`
: The labs of legend. Default is `c('Gen','Env')`.

`size.text.gen, size.text.env, size.text.lab`
: The size of the text for genotypes, environments and labels, respectively.

`size.line`
: The size of the line in biplots (Both for segments and circles).

`large_label`
: The text size to use for larger labels where `type = 3`, used for the outermost genotypes and where `type = 9`, used for the two selected genotypes. Defaults to 4.5

`axis_expand`
: multiplication factor to expand the axis limits by to enable fitting of labels. Defaults to 1.2

`title`
: Logical values (Defaults to `TRUE`) to include automatically generated informations in the plot such as singular value partitioning, scaling and centering.

`plot_theme`
: The graphical theme of the plot. Default is `plot_theme = theme_metan()`. For more details, see `theme`.

`...`
: Currently not used.

Value

A ggplot2-based biplot.

An object of class `gg,ggplot`.

Author(s)

Tiago Olivoto `<tiagoolivoto@gmail.com>`

References

Yan, W., and M.S. Kang. 2003. GGE biplot analysis: a graphical tool for breeders, geneticists, and agronomists. CRC Press.

Examples

```
library(metan)
mod <- gge(data_ge, ENV, GEN, GY)
plot(mod)
plot(mod,
     type = 2,
     col.gen = 'blue',
     col.env = 'red',
```

```
        size.text.gen = 2)
```

---

plot.mtsi *Plot the multi-trait stability index*

Description

Makes a radar plot showing the multitrait stability index proposed by Olivoto et al. (2019)

Usage

```
## S3 method for class 'mtsi'
plot(
  x,
  SI = 15,
  radar = TRUE,
  arrange.label = FALSE,
  size.point = 2.5,
  col.sel = "red",
  col.nonsel = "black",
  size.text = 10,
  ...
)
```

Arguments

x	An object of class mtsi
SI	An integer [0-100]. The selection intensity in percentage of the total number of genotypes.
radar	Logical argument. If true (default) a radar plot is generated after using coord_polar().
arrange.label	Logical argument. If TRUE, the labels are arranged to avoid text overlapping. This becomes useful when the number of genotypes is large, say, more than 30.
size.point	The size of the point in graphic. Defaults to 2.5.
col.sel	The colour for selected genotypes.
col.nonsel	The colour for nonselected genotypes.
size.text	The size for the text in the plot. Defaults to 10.
...	Other arguments to be passed from ggplot2::theme().

Value

An object of class gg, ggplot.

Author(s)

Tiago Olivoto <tiagoolivoto@gmail.com>

References

Olivoto, T., A.D.C. L\'ucio, J.A.G. da silva, B.G. Sari, and M.I. Diel. 2019. Mean performance and stability in multi-environment trials II: Selection based on multiple traits. Agron. J. (in press).

Examples

```
library(metan)
mtsi_model <- waasb(data_ge, ENV, GEN, REP, resp = c(GY, HM))
mtsi_index <- mtsi(mtsi_model)
plot(mtsi_index)
```

---

plot.performs_ammi *Several types of residual plots*

Description

Residual plots for a output model of class `performs_ammi`. Seven types of plots are produced: (1) Residuals vs fitted, (2) normal Q-Q plot for the residuals, (3) scale-location plot (standardized residuals vs Fitted Values), (4) standardized residuals vs Factor-levels, (5) Histogram of raw residuals and (6) standardized residuals vs observation order, and (7) 1:1 line plot

Usage

```
## S3 method for class 'performs_ammi'
plot(
  x,
  var = 1,
  conf = 0.95,
  labels = FALSE,
  plot_theme = theme_metan(),
  band.alpha = 0.2,
  point.alpha = 0.8,
  fill.hist = "gray",
  col.hist = "black",
  col.point = "black",
  col.line = "red",
  col.lab.out = "red",
  size.lab.out = 2.5,
  size.tex.lab = 10,
  size.shape = 1.5,
  bins = 30,
  which = c(1:4),
  ncol = NULL,
```

```
  nrow = NULL,
  ...
)
```

Arguments

x	An object of class performs_ammi.
var	The variable to plot. Defaults to var = 1 the first variable of x.
conf	Level of confidence interval to use in the Q-Q plot (0.95 by default).
labels	Logical argument. If TRUE labels the points outside confidence interval limits.
plot_theme	The graphical theme of the plot. Default is plot_theme = theme_metan(). For more details, see theme.
band.alpha, point.alpha	The transparency of confidence band in the Q-Q plot and the points, respectively. Must be a number between 0 (opaque) and 1 (full transparency).
fill.hist	The color to fill the histogram. Default is 'gray'.
col.hist	The color of the border of the the histogram. Default is 'black'.
col.point	The color of the points in the graphic. Default is 'black'.
col.line	The color of the lines in the graphic. Default is 'red'.
col.lab.out	The color of the labels for the 'outlying' points.
size.lab.out	The size of the labels for the 'outlying' points.
size.tex.lab	The size of the text in axis text and labels.
size.shape	The size of the shape in the plots.
bins	The number of bins to use in the histogram. Default is 30.
which	Which graphics should be plotted. Default is which = c(1:4) that means that the first four graphics will be plotted.
ncol, nrow	The number of columns and rows of the plot pannel. Defaults to NULL
...	Additional arguments passed on to the function plot_grid

Author(s)

Tiago Olivoto <tiagoolivoto@gmail.com>

Examples

```
library(metan)
model <- performs_ammi(data_ge, ENV, GEN, REP, GY)
model2 <- waas(data_ge, ENV, GEN, REP, GY)
plot(model)
plot(model2,
     which = c(3, 5),
     ncol = 3,
     labels = TRUE,
     size.lab.out = 4)
```

`plot.resp_surf` *Plot the response surface model*

Description

Plot the response surface model using a contour plot

Usage

```
## S3 method for class 'resp_surf'
plot(x, xlab = NULL, ylab = NULL, region = TRUE, resolution = 100, ...)
```

Arguments

`x`	An object of class `resp_surf`
`xlab`	The label for the x axis
`ylab`	The label for the y axis
`region`	Logical argument indicating whether regions between contour lines should be colored.
`resolution`	The resolution of the contour plot. Defaults to 100. higher values produce high-resolution plots but may increase the computation time.
`...`	Other arguments passed from `contourplot` function. See `contourplot` for more details.

Value

An object of class `trellis`.

Author(s)

Tiago Olivoto <tiagoolivoto@gmail.com>

Examples

```
library(metan)
# A small toy example

df <- data.frame(
 expand.grid(x = seq(0, 4, by = 1),
             y = seq(0, 4, by = 1)),
 z = c(10, 11, 12, 11, 10,
       14, 15, 16, 15, 14,
       16, 17, 18, 17, 16,
       14, 15, 16, 15, 14,
       10, 11, 12, 11, 10)
)
```

```
mod <- resp_surf(df, x, y, resp = z)
plot(mod)
```

`plot.waas` *Several types of residual plots*

Description

Residual plots for a output model of class `waas`. Seven types of plots are produced: (1) Residuals vs fitted, (2) normal Q-Q plot for the residuals, (3) scale-location plot (standardized residuals vs Fitted Values), (4) standardized residuals vs Factor-levels, (5) Histogram of raw residuals and (6) standardized residuals vs observation order, and (7) 1:1 line plot

Usage

```
## S3 method for class 'waas'
plot(
  x,
  var = 1,
  conf = 0.95,
  labels = FALSE,
  plot_theme = theme_metan(),
  band.alpha = 0.2,
  point.alpha = 0.8,
  fill.hist = "gray",
  col.hist = "black",
  col.point = "black",
  col.line = "red",
  col.lab.out = "red",
  size.lab.out = 2.5,
  size.tex.lab = 10,
  size.shape = 1.5,
  bins = 30,
  which = c(1:4),
  ncol = NULL,
  nrow = NULL,
  ...
)
```

Arguments

`x`	An object of class `waas`.
`var`	The variable to plot. Defaults to `var = 1` the first variable of `x`.
`conf`	Level of confidence interval to use in the Q-Q plot (0.95 by default).
`labels`	Logical argument. If `TRUE` labels the points outside confidence interval limits.

plot_theme	The graphical theme of the plot. Default is plot_theme = theme_metan(). For more details, see theme.
band.alpha, point.alpha	The transparency of confidence band in the Q-Q plot and the points, respectively. Must be a number between 0 (opaque) and 1 (full transparency).
fill.hist	The color to fill the histogram. Default is 'gray'.
col.hist	The color of the border of the the histogram. Default is 'black'.
col.point	The color of the points in the graphic. Default is 'black'.
col.line	The color of the lines in the graphic. Default is 'red'.
col.lab.out	The color of the labels for the 'outlying' points.
size.lab.out	The size of the labels for the 'outlying' points.
size.tex.lab	The size of the text in axis text and labels.
size.shape	The size of the shape in the plots.
bins	The number of bins to use in the histogram. Default is 30.
which	Which graphics should be plotted. Default is which = c(1:4) that means that the first four graphics will be plotted.
ncol, nrow	The number of columns and rows of the plot pannel. Defaults to NULL
...	Additional arguments passed on to the function plot_grid

Author(s)

Tiago Olivoto <tiagoolivoto@gmail.com>

Examples

```
library(metan)
model <- performs_ammi(data_ge, ENV, GEN, REP, GY)
model2 <- waas(data_ge, ENV, GEN, REP, GY)
plot(model)
plot(model2,
     which = c(3, 5),
     ncol = 2,
     labels = TRUE,
     size.lab.out = 4)
```

---

plot.waasb *Several types of residual plots*

Description

Residual plots for a output model of class waas and waasb. Six types of plots are produced: (1) Residuals vs fitted, (2) normal Q-Q plot for the residuals, (3) scale-location plot (standardized residuals vs Fitted Values), (4) standardized residuals vs Factor-levels, (5) Histogram of raw residuals and (6) standardized residuals vs observation order. For a waasb object, normal Q-Q plot for random effects may also be obtained declaring type = 're'

Usage

```
## S3 method for class 'waasb'
plot(
  x,
  var = 1,
  type = "res",
  conf = 0.95,
  out = "print",
  labels = FALSE,
  plot_theme = theme_metan(),
  alpha = 0.2,
  fill.hist = "gray",
  col.hist = "black",
  col.point = "black",
  col.line = "red",
  col.lab.out = "red",
  size.lab.out = 2.5,
  size.tex.lab = 10,
  size.shape = 1.5,
  bins = 30,
  which = c(1:4),
  ncol = NULL,
  nrow = NULL,
  ...
)
```

Arguments

x	An object of class waasb.
var	The variable to plot. Defaults to var = 1 the first variable of x.
type	If type = 're', normal Q-Q plots for the random effects are obtained.
conf	Level of confidence interval to use in the Q-Q plot (0.95 by default).
out	How the output is returned. Must be one of the 'print' (default) or 'return'.
labels	Logical argument. If TRUE labels the points outside confidence interval limits.
plot_theme	The graphical theme of the plot. Default is plot_theme = theme_metan(). For more details, see theme.
alpha	The transparency of confidence band in the Q-Q plot. Must be a number between 0 (opaque) and 1 (full transparency).
fill.hist	The color to fill the histogram. Default is 'gray'.
col.hist	The color of the border of the the histogram. Default is 'black'.
col.point	The color of the points in the graphic. Default is 'black'.
col.line	The color of the lines in the graphic. Default is 'red'.
col.lab.out	The color of the labels for the 'outlying' points.
size.lab.out	The size of the labels for the 'outlying' points.

size.tex.lab	The size of the text in axis text and labels.
size.shape	The size of the shape in the plots.
bins	The number of bins to use in the histogram. Default is 30.
which	Which graphics should be plotted. Default is which = c(1:4) that means that the first four graphics will be plotted.
ncol, nrow	The number of columns and rows of the plot pannel. Defaults to NULL
...	Additional arguments passed on to the function plot_grid

Author(s)

Tiago Olivoto <tiagoolivoto@gmail.com>

Examples

```
library(metan)
model2 <- waasb(data_ge,
                resp = GY,
                gen = GEN,
                env = ENV,
                rep = REP)
plot(model2)
```

---

plot.wsmp *Plot heat maps with genotype ranking*

Description

Plot heat maps with genotype ranking in two ways.

Usage

```
## S3 method for class 'wsmp'
plot(
  x,
  var = 1,
  type = 2,
  export = FALSE,
  file.type = "pdf",
  file.name = NULL,
  width = 6,
  height = 5,
  size.lab = 1,
  margins = c(5, 4),
  y.lab = NULL,
```

```
  x.lab = NULL,
  key.lab = "Genotype ranking",
  resolution = 300,
  ...
)
```

Arguments

x	The object returned by the function wsmp.
var	The variable to plot. Defaults to var = 1 the first variable of x.
type	1 = Heat map Ranks: this graphic shows the genotype ranking considering the WAAS estimated with different numbers of Principal Components; 2 = Heat map WAASY-GY ratio: this graphic shows the genotype ranking considering the different combinations in the WAAS/GY ratio.
export	Export (or not) the plot. Default is FALSE.
file.type	If export = TRUE define the type of file to be exported. Default is pdf, Graphic can also be exported in *.tiff format by declaring file.type = 'tiff'.
file.name	The name of the file for exportation, default is NULL, i.e. the files are automatically named.
width	The width 'inch' of the plot. Default is 8.
height	The height 'inch' of the plot. Default is 7.
size.lab	The label size of the plot. It is suggested attribute 1
margins	Numeric vector of length 2 containing the margins for column and row names, respectively. Default is c(5,4).
y.lab	The label of y axis. Default is 'Genotypes'.
x.lab	The label of x axis. Default is 'Number of axes'.
key.lab	The label of color key. Default is 'Genotype ranking'.
resolution	Valid parameter if file.type = 'tiff'. Define the resolution of the plot. Default is '300'.
...	Currently not used.

Details

The first type of heatmap shows the genotype ranking depending on the number of principal component axis used for estimating the WAASB index. An euclidean distance-based dendrogram is used for grouping the genotype ranking for both genotypes and principal component axis. The second type of heatmap shows the genotype ranking depending on the WAASB/GY ratio. The ranks obtained with a ratio of 100/0 considers exclusively the stability for the genotype ranking. On the other hand, a ratio of 0/100 considers exclusively the productivity for the genotype ranking. Four clusters are estimated (1) unproductive and unstable genotypes; (2) productive, but unstable genotypes; (3) stable, but unproductive genotypes; and (4), productive and stable genotypes.

Author(s)

Tiago Olivoto <tiagoolivoto@gmail.com>

Examples

```
library(metan)
model <- waas(data_ge2, ENV, GEN, REP, PH) %>%
         wsmp()
plot(model)
```

`plot_blup` *Plot the BLUPs for genotypes*

Description

Plot the predicted BLUP of the genotypes.

Usage

```
plot_blup(
  x,
  var = 1,
  prob = 0.05,
  export = FALSE,
  file.type = "pdf",
  file.name = NULL,
  plot_theme = theme_metan(),
  width = 6,
  height = 6,
  size.err.bar = 0.5,
  size.shape = 3.5,
  size.tex.lab = 12,
  height.err.bar = 0.3,
  x.lim = NULL,
  x.breaks = waiver(),
  col.shape = c("blue", "red"),
  y.lab = "Genotypes",
  x.lab = "Predicted Grain Yield",
  resolution = 300,
  ...
)
```

Arguments

`x`	The `waasb object`
`var`	The variable to plot. Defaults to `var = 1` the first variable of `x`.
`prob`	The probability error for constructing confidence interval.

export	Export (or not) the plot. Default is TRUE.
file.type	If export = TRUE, define the type of file to be exported. Default is pdf, Graphic can also be exported in *.tiff format by declaring file.type = "tiff".
file.name	The name of the file for exportation, default is NULL, i.e. the files are automatically named.
plot_theme	The graphical theme of the plot. Default is plot_theme = theme_metan(). For more details, see theme.
width	The width "inch" of the plot. Default is 6.
height	The height "inch" of the plot. Default is 6.
size.err.bar	The size of the error bar for the plot. Default is 0.5.
size.shape	The size of the shape (both for genotypes and environments). Default is 3.5.
size.tex.lab	The size of the text in axis text and labels.
height.err.bar	The height for error bar. Default is 0.3.
x.lim	The range of x-axis. Default is NULL (maximum and minimum values of the data set). New arguments can be inserted as x.lim = c(x.min, x.max).
x.breaks	The breaks to be plotted in the x-axis. Default is authomatic breaks. New arguments can be inserted as x.breaks = c(breaks)
col.shape	A vector of length 2 that contains the color of shapes for genotypes above and below of the mean, respectively. Default is c("blue", "red").
y.lab	The label of the y-axis in the plot. Default is "Genotypes".
x.lab	The label of the x-axis in the plot. Default is "Predicted Grain Yield".
resolution	The resolution of the plot. Parameter valid if file.type = "tiff" is used. Default is 300 (300 dpi)
...	Currently not used.

Value

An object of class gg, ggplot.

Author(s)

Tiago Olivoto <tiagoolivoto@gmail.com>

See Also

plot_scores, plot_waasby

Examples

```
library(metan)
BLUP <- waasb(data_ge,
              resp = c(GY, HM),
              gen = GEN,
              env = ENV,
```

```
              rep = REP)
plot_blup(BLUP)
```

---

plot_ci *Plot the confidence interval for correlation*

Description

This function plots the 95 correlation coefficient generated by the function `corr_ci`.

Usage

```
plot_ci(
  object,
  x.lab = NULL,
  y.lab = NULL,
  y.lim = NULL,
  y.breaks = waiver(),
  shape = 21,
  col.shape = "black",
  fill.shape = "orange",
  size.shape = 2.5,
  width.errbar = 0.5,
  main = TRUE,
  invert.axis = TRUE,
  plot_theme = theme_metan()
)
```

Arguments

`object`	An object generate by the function `corr_ci()`
`x.lab`	The label of x-axis, set to 'Pairwise combinations'. New arguments can be inserted as `x.lab = 'my label'`.
`y.lab`	The label of y-axis, set to 'Pearson's correlation coefficient' New arguments can be inserted as `y.lab = 'my label'`.
`y.lim`	The range of x-axis. Default is `NULL`. The same arguments than `x.lim` can be used.
`y.breaks`	The breaks to be plotted in the x-axis. Default is `authomatic breaks`. The same arguments than `x.breaks` can be used.
`shape`	The shape point to represent the correlation coefficient. Default is `21` (circle). Values must be between `21-25`: `21` (circle), `22` (square), `23` (diamond), `24` (up triangle), and `25` (low triangle).

col.shape	The color for the shape edge. Set to black.
fill.shape	The color to fill the shape. Set to orange.
size.shape	The size for the shape point. Set to 2.5.
width.errbar	The width for the errorbar showing the CI.
main	The title of the plot. Set to main = FALSE to ommite the plot title.
invert.axis	Should the names of the pairwise correlation appear in the y-axis?
plot_theme	The graphical theme of the plot. Default is plot_theme = theme_metan(). For more details, see theme.

Value

An object of class gg,ggplot.

Examples

```
library(metan)
library(dplyr)

data_ge2 %>%
select(contains('E')) %>%
corr_ci() %>%
plot_ci()
```

---

plot_eigen *Plot the eigenvalues*

Description

Plot the eigenvalues for from singular value decomposition of BLUP interaction effects matrix.

Usage

```
plot_eigen(
  x,
  var = 1,
  export = FALSE,
  plot_theme = theme_metan(),
  file.type = "pdf",
  file.name = NULL,
  width = 6,
  height = 6,
  size.shape = 3.5,
  size.line = 1,
  size.tex.lab = 12,
  y.lab = "Eigenvalue",
```

```
  y2.lab = "Accumulated variance",
  x.lab = "Number of multiplicative terms",
  resolution = 300,
  ...
)
```

Arguments

x	The waasb object
var	The variable to plot. Defaults to var = 1 the first variable of x.
export	Export (or not) the plot. Default is TRUE.
plot_theme	The graphical theme of the plot. Default is plot_theme = theme_metan(). For more details, see theme.
file.type	If export = TRUE, define the type of file to be exported. Default is pdf, Graphic can also be exported in *.tiff format by declaring file.type = "tiff".
file.name	The name of the file for exportation, default is NULL, i.e. the files are automatically named.
width	The width "inch" of the plot. Default is 6.
height	The height "inch" of the plot. Default is 6.
size.shape	The size of the shape. Default is 3.5.
size.line	The size of the line. Default is 1.
size.tex.lab	The size of the text in axis text and labels.
y.lab	The label of the y-axis in the plot. Default is "Eigenvalue".
y2.lab	The label of the second y-axis in the plot. Default is "Accumulated variance".
x.lab	The label of the x-axis in the plot. Default is "Number of multiplicative terms".
resolution	The resolution of the plot. Parameter valid if file.type = "tiff" is used. Default is 300 (300 dpi)
...	Currently not used.

Value

An object of class gg, ggplot.

Author(s)

Tiago Olivoto <tiagoolivoto@gmail.com>

See Also

plot_scores, plot_waasby

Examples

```
library(metan)
BLUP <- waasb(data_ge,
              resp = c(GY, HM),
              gen = GEN,
              env = ENV,
              rep = REP)
plot_eigen(BLUP)
```

`plot_factbars` *Fast way to create a bar plot*

Description

Create a bar plot based on categorical (one or two) variables and one numeric variable.

Usage

```
plot_factbars(
  .data,
  ...,
  resp,
  y.expand = 1,
  y.breaks = waiver(),
  xlab = NULL,
  ylab = NULL,
  lab.bar = NULL,
  lab.bar.hjust = 0.5,
  lab.bar.vjust = -0.5,
  lab.bar.angle = 0,
  size.text.bar = 5,
  lab.x.hjust = 0.5,
  lab.x.vjust = 1,
  lab.x.angle = 0,
  errorbar = TRUE,
  stat.erbar = "se",
  width.erbar = 0.3,
  level = 0.95,
  invert = FALSE,
  col = TRUE,
  palette = "Spectral",
  width.bar = 0.9,
  legend.position = "bottom",
```

```
  size.text = 12,
  fontfam = "sans",
  na.rm = TRUE,
  verbose = FALSE
)
```

Arguments

.data	The data set
...	A comma-separated list of unquoted variable names. Must be up to two variables.
resp	The response variable
y.expand	A multiplication factor to expand the y axis.. Defaults to 1.
y.breaks	The breaks to be plotted in the y-axis. Defaults to waiver(). authomatic breaks. The same arguments than x.breaks can be used.
xlab	The x label
ylab	The y label
lab.bar	A vector of characters to show in each bar. Defaults to NULL.
lab.bar.hjust, lab.bar.vjust	The horizontal and vertical adjust for the labels in the bar. Defaults to 0.5 and -0.5, respectively.
lab.bar.angle	The angle for the labels in the plot. Defaults to 0. Use in combination with lab.bar.hjust and lab.bar.vjust to best fit the labels in the plot.
size.text.bar	The size of the text in the bar labels.
lab.x.hjust, lab.x.vjust	The horizontal and vertical adjust for the labels in the bar. Defaults to 0.5 and 1, respectively.
lab.x.angle	The angle for the labels in x axis. Defaults to 0. Use in combination with lab.x.hjust and lab.x.vjust to best fit the labels in the axis.
errorbar	Logical argument, set to TRUE. In this case, an error bar is shown.
stat.erbar	The statistic to be shown in the errorbar. Must be one of the stat.erbar = "se" (standard error, default), stat.erbar = "sd" (standard deviation), or stat.erbar = "ci" (confidence interval), based on the confidence level in the argument level.
width.erbar	The width of the error bar.
level	The confidence level
invert	Logical argument. If TRUE, the order of the factors entered in changes in the graph
col	Logical argument. If FALSE, a gray scale is used.
palette	The color palette to be used. For more details, see ?scale_colour_brewer
width.bar	The width of the bars in the graph. Defaults to 0.9 possible values [0-1].
legend.position	The position of the legend in the plot.

size.text	The size of the text
fontfam	The family of the font text
na.rm	Should 'NA' values be removed to compute the statistics? Defaults to true
verbose	Logical argument. If TRUE a tibble containing the mean, N, standard deviation, standard error of mean and confidence interval is returned.

Value

An object of class gg,ggplot.

See Also

plot_lines, plot_factlines

Examples

```
library(metan)
plot_factbars(data_ge2,
              GEN,
              ENV,
              resp = PH)
```

---

plot_factlines *Fast way to create a line plot*

Description

Create a graphic to show a fitted line based on numerical variables and one grouping variable.

Usage

```
plot_factlines(
  .data,
  x,
  y,
  group,
  fit,
  level = 0.95,
  confidence = TRUE,
  xlab = NULL,
  ylab = NULL,
  legend.position = "bottom",
  grid = FALSE,
  scales = "free",
  col = TRUE,
  alpha = 0.2,
```

```
  size.shape = 1.5,
  size.line = 1,
  size.text = 12,
  fontfam = "sans",
  plot_theme = theme_metan()
)
```

Arguments

.data	The data set
x	The variable in data to be shown in the x axis
y	The variable in data to be shown in the y axis
group	The grouping variable
fit	The polynomial degree to use. It must be an integer between 1 (linear fit) to 4 (fourth-order polynomial regression.), or a numeric vector with the same length of the variable in group
level	The fonfidence level
confidence	Display confidence interval around smooth? (TRUE by default)
xlab	The x label
ylab	The y label
legend.position	The position of the legend. Defaults to 'bottom'.
grid	Logical argument. If TRUE then a grid will be created.
scales	If grid = TRUE scales controls how the scales are in the plot. Possible values are 'free' (default), 'fixed', 'free_x' or 'free_y'.
col	The colour to be used in the line plot and points
alpha	The alpha for the color in confidence band
size.shape	The size for the shape in plot
size.line	The size for the line in the plot
size.text	The size of the text
fontfam	The family of the font text
plot_theme	The graphical theme of the plot. Default is plot_theme = theme_metan(). For more details, see theme.

Value

An object of class gg, ggplot.

See Also

plot_lines, plot_factbars

plot_lines *Fast way to create a line plot*

Description

Create a graphic with fitted line based on numerical variables.

Usage

```
plot_lines(
  .data,
  x,
  y,
  fit,
  level = 0.95,
  xlab = NULL,
  ylab = NULL,
  col = "red",
  alpha = 0.2,
  size.shape = 1.5,
  size.line = 1,
  size.text = 12,
  fontfam = "sans"
)
```

Arguments

.data	The data set
x	The variable in data to be shown in the x axis
y	The variable in data to be shown in the y axis
fit	The polynomial degree to use. It must be between 1 (linear fit) to 4 (fourth-order polynomial regression.)
level	The fonfidence level
xlab	The x lab
ylab	The y lab
col	The colour to be used in the line plot and points
alpha	The alpha for the color in confidence band
size.shape	The size for the shape in plot
size.line	The size for the line in the plot
size.text	The size of the text
fontfam	The family of the font text

Value

An object of class gg,ggplot.

See Also

plot_factbars plot_factlines

---

plot_scores *Plot scores in different graphical interpretations*

Description

Plot scores of genotypes and environments in different graphical interpretations.

Usage

```
plot_scores(
  x,
  var = 1,
  type = 1,
  polygon = FALSE,
  title = TRUE,
  plot_theme = theme_metan(),
  axis.expand = 1.1,
  x.lim = NULL,
  y.lim = NULL,
  x.breaks = waiver(),
  y.breaks = waiver(),
  x.lab = NULL,
  y.lab = NULL,
  shape.gen = 21,
  shape.env = 23,
  size.shape = 2.2,
  size.bor.tick = 0.3,
  size.tex.lab = 12,
  size.tex.pa = 3.5,
  size.line = 0.5,
  size.segm.line = 0.5,
  col.bor.gen = "black",
  col.bor.env = "black",
  col.line = "black",
  col.gen = "blue",
  col.env = "forestgreen",
  col.alpha.gen = 0.9,
  col.alpha.env = 0.9,
  col.segm.gen = transparent_color(),
  col.segm.env = "forestgreen",
```

```
  repulsion = 1,
  leg.lab = c("Env", "Gen"),
  line.type = "solid",
  line.alpha = 0.9,
  resolution = 300,
  file.type = "pdf",
  export = FALSE,
  file.name = NULL,
  width = 8,
  height = 7,
  color = TRUE,
  ...
)
```

Arguments

x	An object fitted with the functions performs_ammi, waas or waasb.
var	The variable to plot. Defaults to var = 1 the first variable of x.
type	type of biplot to produce • type = 1 Produces an AMMI1 biplot (Y x PC1) to make inferences related to stability and productivity. • type = 2 The default, produces an AMMI2 biplot (PC1 x PC2) to make inferences related to the interaction effects. • type = 3 Valid for objects of class waas or waasb, produces a biplot showing the GY x WAASB. • type = 4 Produces a plot with the Nominal yield x Environment PC.
polygon	Logical argument. If TRUE, a polygon is drawn when type = 2.
title	Logical values (Defaults to TRUE) to include automatically generated titles
plot_theme	The graphical theme of the plot. Default is plot_theme = theme_metan(). For more details, see theme.
axis.expand	Multiplication factor to expand the axis limits by to enable fitting of labels. Default is 1.1.
x.lim, y.lim	The range of x and y axes, respectively. Default is NULL (maximum and minimum values of the data set). New values can be inserted as x.lim = c(x.min,x.max) or y.lim = c(y.min,y.max).
x.breaks, y.breaks	The breaks to be plotted in the x and y axes, respectively. Defaults to waiver() (automatic breaks). New values can be inserted, for example, as x.breaks = c(0.1,0.2,0.3) or x.breaks = seq(0,1,by = 0.2)
x.lab, y.lab	The label of x and y axes, respectively. Defaults to NULL, i.e., each plot has a default axis label. New values can be inserted as x.lab = 'my label'.
shape.gen, shape.env	The shape for genotypes and environments indication in the biplot. Default is 21 (circle) for genotypes and 23 (diamond) for environments. Values must be between 21-25: 21 (circle), 22 (square), 23 (diamond), 24 (up triangle), and 25 (low triangle).

size.shape The size of the shape (both for genotypes and environments). Default is 2.2.

size.bor.tick The size of tick of shape. Default is 0.3. The size of the shape will be size.shape + size.bor.tick

size.tex.lab, size.tex.pa
The size of the text for labels (Defaults to 12) and plot area (Defaults to 3.5), respectively.

size.line The size of the line that indicate the means in the biplot. Default is 0.5.

size.segm.line The size of the segment that start in the origin of the biplot and end in the scores values. Default is 0.5.

col.bor.gen, col.bor.env
The color of the shape's border for genotypes and environments, respectively.

col.line The color of the line that indicate the means in the biplot. Default is 'gray'

col.gen, col.env
The shape color for genotypes (Defaults to 'blue') and environments ('forestgreen'). Must be length one or a vector of colors with the same length of the number of genotypes/environments.

col.alpha.gen, col.alpha.env
The alpha value for the color for genotypes and environments, respectively. Default is 0.9. Values must be between 0 (full transparency) to 1 (full color).

col.segm.gen, col.segm.env
The color of segment for genotypes (Defaults to transparent_color()) and environments (Defaults to 'forestgreen'), respectively. Valid arguments for plots with type = 1 or type = 2 graphics.

repulsion Force of repulsion between overlapping text labels. Defaults to 1.

leg.lab The labs of legend. Default is Gen and Env.

line.type The type of the line that indicate the means in the biplot. Default is 'solid'. Other values that can be attributed are: 'blank', no lines in the biplot, 'dashed','dotted','dotdash', 'twodash'.

line.alpha The alpha value that combine the line with the background to create the appearance of partial or full transparency. Default is 0.4. Values must be between '0' (full transparency) to '1' (full color).

resolution The resolution of the plot. Parameter valid if file.type = 'tiff' is used. Default is 300 (300 dpi)

file.type The type of file to be exported. Valid parameter if export = T|TRUE. Default is 'pdf'. The graphic can also be exported in *.tiff format by declaring file.type = 'tiff'.

export Export (or not) the plot. Default is FALSE.

file.name The name of the file for exportation, default is NULL, i.e. the files are automatically named.

width The width 'inch' of the plot. Default is 8.

height The height 'inch' of the plot. Default is 7.

color Should type 4 plot have colors? Default to TRUE.

... Currently not used.

Details

Biplots type 1 and 2 are well known in AMMI analysis. In the plot type 3, the scores of both genotypes and environments are plotted considering the response variable and the WAASB, an stability index that considers all significant principal component axis of traditional AMMI models or all principal component axis estimated with BLUP-interaction effects (Olivoto et al. 2019). Plot type 4 may be used to better understand the well known 'which-won-where' pattern, facilitating the recommendation of appropriate genotypes targeted for specific environments, thus allowing the exploitation of narrow adaptations.

Value

An object of class `gg,ggplot`.

Author(s)

Tiago Olivoto <tiagoolivoto@gmail.com>

References

Olivoto, T., A.D.C. L\'ucio, J.A.G. da silva, V.S. Marchioro, V.Q. de Souza, and E. Jost. 2019. Mean performance and stability in multi-environment trials I: Combining features of AMMI and BLUP techniques. Agron. J. 111:2949-2960. doi:10.2134/agronj2019.03.0220

See Also

plot_eigen

Examples

```
library(metan)
# AMMI model
ammi_model = performs_ammi(data_ge, ENV, GEN, REP,
                           resp = c(GY, HM))

# GY x PC1 (variable GY)
plot_scores(ammi_model,
            col.env = 'olivedrab',
            col.gen = 'orange2',
            x.lab = 'My own x label')

# PC1 x PC2 (variable HM)
plot_scores(ammi_model,
            var = "HM", # or var = 2
            type = 2,
            polygon = TRUE)

# PC1 x PC2 (variable HM)
# Draw a convex hull polygon
plot_scores(ammi_model,
            var = "HM",
```

```
            type = 2,
            polygon = TRUE)

# WAASB index
waasb_model = waasb(data_ge, ENV, GEN, REP, GY)

# GY x WAASB
plot_scores(waasb_model,
            type = 3,
            size.tex.pa = 2,
            size.tex.lab = 16)
```

`plot_waasby` *Plot WAASBY values for genotype ranking*

Description

Plot heat maps with genotype ranking in two ways.

Usage

```
plot_waasby(
  x,
  var = 1,
  export = F,
  file.type = "pdf",
  file.name = NULL,
  plot_theme = theme_metan(),
  width = 6,
  height = 6,
  size.shape = 3.5,
  size.tex.lab = 12,
  col.shape = c("blue", "red"),
  x.lab = "WAASBY",
  y.lab = "Genotypes",
  x.breaks = waiver(),
  resolution = 300,
  ...
)
```

Arguments

`x`	The `WAASBY object`
`var`	The variable to plot. Defaults to `var = 1` the first variable of `x`.
`export`	Export (or not) the plot. Default is `T`.
`file.type`	The type of file to be exported. Default is `pdf`, Graphic can also be exported in `*.tiff` format by declaring `file.type = "tiff"`.

file.name	The name of the file for exportation, default is NULL, i.e. the files are automatically named.
plot_theme	The graphical theme of the plot. Default is plot_theme = theme_metan(). For more details, see theme.
width	The width "inch" of the plot. Default is 8.
height	The height "inch" of the plot. Default is 7.
size.shape	The size of the shape in the plot. Default is 3.5.
size.tex.lab	The size of the text in axis text and labels.
col.shape	A vector of length 2 that contains the color of shapes for genotypes above and below of the mean, respectively. Default is c("blue","red").
x.lab	The label of the x axis in the plot. Default is "WAASBY".
y.lab	The label of the y axis in the plot. Default is "Genotypes".
x.breaks	The breaks to be plotted in the x-axis. Default is authomatic breaks. New arguments can be inserted as x.breaks = c(breaks)
resolution	The resolution of the plot. Parameter valid if file.type = "tiff" is used. Default is 300 (300 dpi)
...	Currently not used.

Value

An object of class gg,ggplot.

Author(s)

Tiago Olivoto <tiagoolivoto@gmail.com>

See Also

plot_scores

Examples

```
library(metan)
library(ggplot2)
waasby <- waasb(data_ge,
                resp = GY,
                gen = GEN,
                env = ENV,
                rep = REP)
waasby2 <- waas(data_ge,
                resp = GY,
                gen = GEN,
                env = ENV,
                rep = REP)
plot_waasby(waasby)
plot_waasby(waasby2) +
```

```
theme_gray() +
theme(legend.position = "bottom",
      legend.background = element_blank(),
      legend.title = element_blank(),
      legend.direction = "horizontal")
```

---

predict.gge *Predict a two-way table based on GGE model*

Description

Predict the means for a genotype-vs-environment trial based on a Genotype plus Genotype-vs-Environment interaction (GGE) model.

Usage

```
## S3 method for class 'gge'
predict(object, naxis = 2, output = "wide", ...)
```

Arguments

object	An object of class gge.
naxis	The the number of principal components to be used in the prediction. Generally, two axis may be used. In this case, the estimated values will be those shown in the biplot.
output	The type of output. It must be one of the 'long' (default) returning a long-format table with the columns for environment (ENV), genotypes (GEN) and response variable (Y); or 'wide' to return a two-way table with genotypes in the row, environments in the columns, filled by the estimated values.
...	Additional parameter for the function

Details

This function is used to predict the response variable of a two-way table (for examples the yielding of g genotypes in e environments) based on GGE model. This prediction is based on the number of principal components used. For more details see Yan and Kang (2007).

Value

A two-way table with genotypes in rows and environments in columns if output = "wide" or a long format (columns ENV, GEN and Y) if output = "long" with the predicted values by the GGE model.

Author(s)

Tiago Olivoto <tiagoolivoto@gmail.com>

References

Yan, W., and M.S. Kang. 2003. GGE biplot analysis: a graphical tool for breeders, geneticists, and agronomists. CRC Press.

Examples

```
library(metan)
mod <- gge(data_ge, GEN, ENV, c(GY, HM))
predict(mod)
```

---

predict.performs_ammi *Predict the means of a performs_ammi object*

Description

Predict the means of a performs_ammi object considering a specific number of axis.

Usage

```
## S3 method for class 'performs_ammi'
predict(object, naxis = 2, ...)
```

Arguments

object	An object of class performs_ammi
naxis	The the number of axis to be use in the prediction. If object has more than one variable, then naxis must be a vector.
...	Additional parameter for the function

Details

This function is used to predict the response variable of a two-way table (for examples the yielding of the i-th genotype in the j-th environment) based on AMMI model. This prediction is based on the number of multiplicative terms used. If naxis = 0, only the main effects (AMMI0) are used. In this case, the predicted mean will be the predicted value from OLS estimation. If naxis = 1 the AMMI1 (with one multiplicative term) is used for predicting the response variable. If naxis = min(gen-1;env-1), the AMMIF is fitted and the predicted value will be the cell mean, i.e. the mean of R-replicates of the i-th genotype in the j-th environment. The number of axis to be used must be carefully chosen. Procedures based on Postdictive success (such as Gollobs's d.f.) or Predictive success (such as cross-validation) should be used to do this. This package provide both. performs_ammi function compute traditional AMMI analysis showing the number of significant axis. On the other hand, cv_ammif function provide a cross-validation, estimating the RMSPD of all AMMI-family models, based on resampling procedures.

Value

A list where each element is the predicted values by the AMMI model for each variable.

Author(s)

Tiago Olivoto <tiagoolivoto@gmail.com>

Examples

```
library(metan)
model <- performs_ammi(data_ge, ENV, GEN, REP,
                       resp = c(GY, HM))
# Predict GY with 3 IPCA and HM with 1 IPCA
predict <- predict(model, naxis = c(3, 1))
```

---

predict.waas *Predict the means of a waas object*

Description

Predict the means of a waas object considering a specific number of axis.

Usage

```
## S3 method for class 'waas'
predict(object, naxis = 2, ...)
```

Arguments

object	An object of class waas
naxis	The the number of axis to be use in the prediction. If object has more than one variable, then naxis must be a vector.
...	Additional parameter for the function

Details

This function is used to predict the response variable of a two-way table (for examples the yielding of the i-th genotype in the j-th environment) based on AMMI model. This prediction is based on the number of multiplicative terms used. If naxis = 0, only the main effects (AMMI0) are used. In this case, the predicted mean will be the predicted value from OLS estimation. If naxis = 1 the AMMI1 (with one multiplicative term) is used for predicting the response variable. If naxis = min(gen-1;env-1), the AMMIF is fitted and the predicted value will be the cell mean, i.e. the mean of R-replicates of the i-th genotype in the j-th environment. The number of axis to be used must be carefully chosen. Procedures based on Postdictive success (such as Gollobs's d.f.) or Predictive sucess (such as cross-validation) should be used to do this. This package provide both.

waas function compute traditional AMMI analysis showing the number of significant axis. On the other hand, cv_ammif function provide a cross-validation, estimating the RMSPD of all AMMI-family models, based on resampling procedures.

Value

A list where each element is the predicted values by the AMMI model for each variable.

Author(s)

Tiago Olivoto <tiagoolivoto@gmail.com>

Examples

```
library(metan)
model = waas(data_ge,
             env = ENV,
             gen = GEN,
             rep = REP,
             resp = c(GY, HM))
# Predict GY with 3 IPCA and HM with 1 IPCA
predict = predict(model, naxis = c(3, 1))
```

---

print.AMMI_indexes *Print an object of class AMMI_indexes*

Description

Print the AMMI_indexes object in two ways. By default, the results are shown in the R console. The results can also be exported to the directory into a *.txt file.

Usage

```
## S3 method for class 'AMMI_indexes'
print(x, export = FALSE, file.name = NULL, digits = 3, ...)
```

Arguments

x	An object of class AMMI_indexes.
export	A logical argument. If TRUE, a *.txt file is exported to the working directory.
file.name	The name of the file if export = TRUE
digits	The significant digits to be shown.
...	Options used by the tibble package to format the output. See tibble::print() for more details.

Author(s)

Tiago Olivoto <tiagoolivoto@gmail.com>

Examples

```
library(metan)
model <- performs_ammi(data_ge, ENV, GEN, REP, GY) %>%
         AMMI_indexes()
print(model)
```

---

print.Annicchiarico *Print an object of class Annicchiarico*

Description

Print the Annicchiarico object in two ways. By default, the results are shown in the R console. The results can also be exported to the directory into a *.txt file.

Usage

```
## S3 method for class 'Annicchiarico'
print(x, export = FALSE, file.name = NULL, digits = 3, ...)
```

Arguments

x	The Annicchiarico x
export	A logical argument. If TRUE, a *.txt file is exported to the working directory.
file.name	The name of the file if export = TRUE
digits	The significant digits to be shown.
...	Options used by the tibble package to format the output. See tibble::print() for more details.

Author(s)

Tiago Olivoto <tiagoolivoto@gmail.com>

Examples

```
library(metan)
Ann <- Annicchiarico(data_ge2,
  env = ENV,
  gen = GEN,
  rep = REP,
  resp = PH
)
print(Ann)
```

print.anova_ind *Print an object of class anova_ind*

Description

Print the anova_ind object in two ways. By default, the results are shown in the R console. The results can also be exported to the directory into a *.txt file.

Usage

```
## S3 method for class 'anova_ind'
print(x, export = FALSE, file.name = NULL, digits = 3, ...)
```

Arguments

x	An object of class anova_ind.
export	A logical argument. If TRUE, a *.txt file is exported to the working directory.
file.name	The name of the file if export = TRUE
digits	The significant digits to be shown.
...	Options used by the tibble package to format the output. See tibble::print() for more details.

Author(s)

Tiago Olivoto <tiagoolivoto@gmail.com>

Examples

```
library(metan)
model <- data_ge %>% anova_ind(ENV, GEN, REP, c(GY, HM))
print(model)
```

print.anova_joint *Print an object of class anova_joint*

Description

Print the anova_joint object in two ways. By default, the results are shown in the R console. The results can also be exported to the directory into a *.txt file.

Usage

```
## S3 method for class 'anova_joint'
print(x, export = FALSE, file.name = NULL, digits = 3, ...)
```

Arguments

x	An object of class anova_joint.
export	A logical argument. If TRUE, a *.txt file is exported to the working directory.
file.name	The name of the file if export = TRUE
digits	The significant digits to be shown.
...	Options used by the tibble package to format the output. See tibble::print() for more details.

Author(s)

Tiago Olivoto <tiagoolivoto@gmail.com>

Examples

```
library(metan)
model <- data_ge %>% anova_joint(ENV, GEN, REP, c(GY, HM))
print(model)
```

---

print.can_cor *Print an object of class can_cor*

Description

Print an object of class can_cor object in two ways. By default, the results are shown in the R console. The results can also be exported to the directory.

Usage

```
## S3 method for class 'can_cor'
print(x, export = FALSE, file.name = NULL, digits = 3, ...)
```

Arguments

x	An object of class can_cor.
export	A logical argument. If TRUE|T, a *.txt file is exported to the working directory
file.name	The name of the file if export = TRUE
digits	The significant digits to be shown.
...	Options used by the tibble package to format the output. See tibble::print() for more details.

Author(s)

Tiago Olivoto <tiagoolivoto@gmail.com>

Examples

```
library(metan)
cc <- can_corr(data_ge2,
  FG = c(PH, EH, EP),
  SG = c(EL, CL, CD, CW, KW, NR, TKW),
  verbose = FALSE
)
print(cc)
```

`print.corr_coef` *Print an object of class corr_coef*

Description

Print the `corr_coef` object in two ways. By default, the results are shown in the R console. The results can also be exported to the directory into a *.txt file.

Usage

```
## S3 method for class 'corr_coef'
print(x, export = FALSE, file.name = NULL, digits = 3, ...)
```

Arguments

`x`	An object of class `corr_coef`
`export`	A logical argument. If `TRUE`, a *.txt file is exported to the working directory.
`file.name`	The name of the file if `export = TRUE`
`digits`	The significant digits to be shown.
`...`	Options used by the tibble package to format the output. See `formatting` for more details.

Author(s)

Tiago Olivoto <tiagoolivoto@gmail.com>

Examples

```
library(metan)
sel <- corr_coef(data_ge2, EP, EL, CD, CL)
print(sel)
```

`print.ecovalence` *Print an object of class ecovalence*

Description

Print the `ecovalence` object in two ways. By default, the results are shown in the R console. The results can also be exported to the directory into a *.txt file.

Usage

```
## S3 method for class 'ecovalence'
print(x, export = FALSE, file.name = NULL, digits = 3, ...)
```

Arguments

`x`	The `ecovalence` x
`export`	A logical argument. If `TRUE`, a *.txt file is exported to the working directory.
`file.name`	The name of the file if `export = TRUE`
`digits`	The significant digits to be shown.
`...`	Options used by the tibble package to format the output. See `tibble::print()` for more details.

Author(s)

Tiago Olivoto <tiagoolivoto@gmail.com>

Examples

```
library(metan)
eco <- ecovalence(data_ge2,
  env = ENV,
  gen = GEN,
  rep = REP,
  resp = PH
)
print(eco)
```

print.env_dissimilarity *Print an object of class env_dissimilarity*

Description

Print the env_dissimilarity object in two ways. By default, the results are shown in the R console. The results can also be exported to the directory into a *.txt file.

Usage

```
## S3 method for class 'env_dissimilarity'
print(x, export = FALSE, file.name = NULL, digits = 3, ...)
```

Arguments

x	An object of class env_dissimilarity.
export	A logical argument. If TRUE, a *.txt file is exported to the working directory.
file.name	The name of the file if export = TRUE
digits	The significant digits to be shown.
...	Currently not used.

Author(s)

Tiago Olivoto <tiagoolivoto@gmail.com>

Examples

```
library(metan)
mod <- env_dissimilarity(data_ge, ENV, GEN, REP, GY)
print(mod)
```

print.Fox *Print an object of class Fox*

Description

Print the Fox object in two ways. By default, the results are shown in the R console. The results can also be exported to the directory into a *.txt file.

Usage

```
## S3 method for class 'Fox'
print(x, export = FALSE, file.name = NULL, digits = 3, ...)
```

Arguments

x	The Fox x
export	A logical argument. If TRUE, a *.txt file is exported to the working directory.
file.name	The name of the file if export = TRUE
digits	The significant digits to be shown.
...	Options used by the tibble package to format the output. See tibble::print() for more details.

Author(s)

Tiago Olivoto <tiagoolivoto@gmail.com>

Examples

```
library(metan)
fox = Fox(data_ge2,
          env = ENV,
          gen = GEN,
          rep = REP,
          resp = PH)
print(fox)
```

---

print.gamem *Print an object of class gamem*

Description

Print the gamem object in two ways. By default, the results are shown in the R console. The results can also be exported to the directory.

Usage

```
## S3 method for class 'gamem'
print(x, export = FALSE, file.name = NULL, digits = 4, ...)
```

Arguments

x	An object fitted with the function gamem .
export	A logical argument. If TRUE, a *.txt file is exported to the working directory
file.name	The name of the file if export = TRUE
digits	The significant digits to be shown.
...	Options used by the tibble package to format the output. See tibble::print() for more details.

Author(s)

Tiago Olivoto <tiagoolivoto@gmail.com>

Examples

```
library(metan)
alpha <- gamem(data_alpha,
  gen = GEN,
  rep = REP,
  block = BLOCK,
  resp = YIELD
)

print(alpha)
```

---

print.ge_factanal *Print an object of class ge_factanal*

Description

Print the ge_factanal object in two ways. By default, the results are shown in the R console. The results can also be exported to the directory.

Usage

```
## S3 method for class 'ge_factanal'
print(x, export = FALSE, file.name = NULL, digits = 4, ...)
```

Arguments

x	An object of class ge_factanal.
export	A logical argument. If TRUE, a *.txt file is exported to the working directory
file.name	The name of the file if export = TRUE
digits	The significant digits to be shown.
...	Options used by the tibble package to format the output. See tibble::print() for more details.

Author(s)

Tiago Olivoto <tiagoolivoto@gmail.com>

Examples

```
model <- ge_factanal(data_ge2,
  env = ENV,
  gen = GEN,
  rep = REP,
  resp = PH
)
print(model)
```

`print.ge_reg` *Print an object of class ge_reg*

Description

Print the `ge_reg` object in two ways. By default, the results are shown in the R console. The results can also be exported to the directory into a *.txt file.

Usage

```
## S3 method for class 'ge_reg'
print(x, export = FALSE, file.name = NULL, digits = 3, ...)
```

Arguments

`x`	An object of class `ge_reg`.
`export`	A logical argument. If `TRUE`, a *.txt file is exported to the working directory.
`file.name`	The name of the file if `export = TRUE`
`digits`	The significant digits to be shown.
`...`	Options used by the tibble package to format the output. See `tibble::print()` for more details.

Author(s)

Tiago Olivoto <tiagoolivoto@gmail.com>

Examples

```
library(metan)
model <- ge_reg(data_ge2, ENV, GEN, REP, PH)
print(model)
```

print.ge_stats *Print an object of class ge_stats*

Description

Print the ge_stats object in two ways. By default, the results are shown in the R console. The results can also be exported to the directory into a *.txt file.

Usage

```
## S3 method for class 'ge_stats'
print(x, what = "all", export = FALSE, file.name = NULL, digits = 3, ...)
```

Arguments

x	An object of class ge_stats.
what	What should be printed. what = "all" for both statistics and ranks, what = "stats" for statistics, and what = "ranks" for ranks.
export	A logical argument. If TRUE, a *.txt file is exported to the working directory.
file.name	The name of the file if export = TRUE
digits	The significant digits to be shown.
...	Options used by the tibble package to format the output. See tibble::print() for more details.

Author(s)

Tiago Olivoto <tiagoolivoto@gmail.com>

Examples

```
library(metan)
model <- ge_stats(data_ge, ENV, GEN, REP, GY)
print(model)
```

print.mtsi *Print an object of class mtsi*

Description

Print a mtsi object in two ways. By default, the results are shown in the R console. The results can also be exported to the directory.

Usage

```
## S3 method for class 'mtsi'
print(x, export = FALSE, file.name = NULL, digits = 4, ...)
```

Arguments

x	An object of class mtsi.
export	A logical argument. If TRUE|T, a *.txt file is exported to the working directory
file.name	The name of the file if export = TRUE
digits	The significant digits to be shown.
...	Options used by the tibble package to format the output. See tibble::print() for more details.

Author(s)

Tiago Olivoto <tiagoolivoto@gmail.com>

Examples

```
library(metan)
# Based on stability only
MTSI_MODEL <- waasb(data_ge,
  resp = c(GY, HM),
  gen = GEN,
  env = ENV,
  rep = REP
)

MTSI_index <- mtsi(MTSI_MODEL)
print(MTSI_index)
```

print.path_coeff *Print an object of class path_coeff*

Description

Print an object generated by the function 'path_coeff()'. By default, the results are shown in the R console. The results can also be exported to the directory.

Usage

```
## S3 method for class 'path_coeff'
print(x, export = FALSE, file.name = NULL, digits = 4, ...)
```

Arguments

x	An object of class path_coeff or group_path.
export	A logical argument. If TRUE, a *.txt file is exported to the working directory
file.name	The name of the file if export = TRUE
digits	The significant digits to be shown.
...	Options used by the tibble package to format the output. See tibble::print() for more details.

Author(s)

Tiago Olivoto <tiagoolivoto@gmail.com>

Examples

```
library(metan)

# KW as dependent trait and all others as predictors
pcoeff <- path_coeff(data_ge2, resp = KW)
print(pcoeff)

# Call the algorithm for selecting a set of predictors
# With minimal multicollinearity (no VIF larger than 5)
pcoeff2 <- path_coeff(data_ge2,
  resp = KW,
  brutstep = TRUE,
  maxvif = 5
)
print(pcoeff2)
```

print.performs_ammi *Print an object of class performs_ammi*

Description

Print the performs_ammi object in two ways. By default, the results are shown in the R console. The results can also be exported to the directory.

Usage

```
## S3 method for class 'performs_ammi'
print(x, export = FALSE, file.name = NULL, digits = 4, ...)
```

Arguments

x	An object of class performs_ammi.
export	A logical argument. If TRUE, a *.txt file is exported to the working directory
file.name	The name of the file if export = TRUE
digits	The significant digits to be shown.
...	Options used by the tibble package to format the output. See tibble::print() for more details.

Author(s)

Tiago Olivoto <tiagoolivoto@gmail.com>

Examples

```
library(metan)
model = performs_ammi(data_ge, ENV, GEN, REP,
                      resp = c(GY, HM))
print(model)
```

print.Schmildt *Print an object of class Schmildt*

Description

Print the Schmildt object in two ways. By default, the results are shown in the R console. The results can also be exported to the directory into a *.txt file.

Usage

```
## S3 method for class 'Schmildt'
print(x, export = FALSE, file.name = NULL, digits = 3, ...)
```

Arguments

x	The Schmildt x
export	A logical argument. If TRUE, a *.txt file is exported to the working directory.
file.name	The name of the file if export = TRUE
digits	The significant digits to be shown.
...	Options used by the tibble package to format the output. See formatting for more details.

Author(s)

Tiago Olivoto <tiagoolivoto@gmail.com>

Examples

```
library(metan)
Sch <- Schmildt(data_ge2,
                env = ENV,
                gen = GEN,
                rep = REP,
                resp = PH)
print(Sch)
```

---

print.Shukla *Print an object of class Shukla*

Description

Print the Shukla object in two ways. By default, the results are shown in the R console. The results can also be exported to the directory into a *.txt file.

Usage

```
## S3 method for class 'Shukla'
print(x, export = FALSE, file.name = NULL, digits = 3, ...)
```

Arguments

x	The Shukla x
export	A logical argument. If TRUE, a *.txt file is exported to the working directory.
file.name	The name of the file if export = TRUE
digits	The significant digits to be shown.
...	Options used by the tibble package to format the output. See tibble::print() for more details.

Author(s)

Tiago Olivoto <tiagoolivoto@gmail.com>

Examples

```
library(metan)
eco <- Shukla(data_ge2,
  env = ENV,
  gen = GEN,
  rep = REP,
  resp = PH
)
print(eco)
```

---

print.superiority *Print an object ofclass* superiority

Description

Print the superiority object in two ways. By default, the results are shown in the R console. The results can also be exported to the directory into a *.txt file.

Usage

```
## S3 method for class 'superiority'
print(x, export = FALSE, file.name = NULL, digits = 3, ...)
```

Arguments

x	An object of class superiority.
export	A logical argument. If TRUE, a *.txt file is exported to the working directory.
file.name	The name of the file if export = TRUE
digits	The significant digits to be shown.
...	Options used by the tibble package to format the output. See tibble::print() for more details.

Author(s)

Tiago Olivoto <tiagoolivoto@gmail.com>

Examples

```
library(metan)
model <- superiority(data_ge2, ENV, GEN, REP, PH)
print(model)
```

print.waas *Print an object of class waas*

Description

Print the waas object in two ways. By default, the results are shown in the R console. The results can also be exported to the directory.

Usage

```
## S3 method for class 'waas'
print(x, export = FALSE, file.name = NULL, digits = 4, ...)
```

Arguments

x	An object of class waas.
export	A logical argument. If TRUE, a *.txt file is exported to the working directory
file.name	The name of the file if export = TRUE
digits	The significant digits to be shown.
...	Options used by the tibble package to format the output. See tibble::print() for more details.

Author(s)

Tiago Olivoto <tiagoolivoto@gmail.com>

Examples

```
library(metan)
model <- waas(data_ge,
  resp = c(GY, HM),
  gen = GEN,
  env = ENV,
  rep = REP
)
print(model)
```

`print.waasb` *Print an object of class waasb*

Description

Print a `waasb` object in two ways. By default, the results are shown in the R console. The results can also be exported to the directory.

Usage

```
## S3 method for class 'waasb'
print(x, export = FALSE, blup = FALSE, file.name = NULL, digits = 4, ...)
```

Arguments

`x`	An object of class `waasb`.
`export`	A logical argument. If `TRUE|T`, a *.txt file is exported to the working directory
`blup`	A logical argument. If `TRUE|T`, the blups are shown.
`file.name`	The name of the file if `export = TRUE`
`digits`	The significant digits to be shown.
`...`	Options used by the tibble package to format the output. See `tibble::print()` for more details.

Author(s)

Tiago Olivoto <tiagoolivoto@gmail.com>

Examples

```
library(metan)
model <- waasb(data_ge,
  resp = c(GY, HM),
  gen = GEN,
  env = ENV,
  rep = REP
)
print(model)
```

print.waas_means *Print an object of class waas_means*

Description

Print the waas_means object in two ways. By default, the results are shown in the R console. The results can also be exported to the directory.

Usage

```
## S3 method for class 'waas_means'
print(x, export = FALSE, file.name = NULL, digits = 4, ...)
```

Arguments

x	An object of class waas_means.
export	A logical argument. If TRUE, a *.txt file is exported to the working directory
file.name	The name of the file if export = TRUE
digits	The significant digits to be shown. See tibble::print() for more details.
...	Currently not used.

Author(s)

Tiago Olivoto <tiagoolivoto@gmail.com>

Examples

```
library(metan)
data_means <- means_by(data_ge, ENV, GEN)
model <- waas_means(data_ge,
                    env = ENV,
                    gen = GEN,
                    resp = everything())
print(model)
```

rbind_fill *Combines data.frames by row filling missing values*

Description

Helper function that combines data.frames by row and fills with . missing values.

Usage

```
rbind_fill(..., fill = ".")
```

Arguments

...	Input dataframes.
fill	What use to fill? Default is "."

Value

A data frame.

Author(s)

Tiago Olivoto <tiagoolivoto@gmail.com>

Examples

```
df1 = data.frame(v1 = c(1, 2), v2 = c(2, 3))
df2 = data.frame(v3 = c(4, 5))
rbind_fill(df1, df2)
rbind_fill(df1, df2, fill = "NA")
```

reorder_cormat *Reorder a correlation matrix*

Description

Reorder the correlation matrix according to the correlation coefficient by using hclust for hierarchical clustering order. This is useful to identify the hidden pattern in the matrix.

Usage

```
reorder_cormat(x)
```

Arguments

x	The correlation matrix

Value

The ordered correlation matrix

Author(s)

Tiago Olivoto <tiagoolivoto@gmail.com>

Examples

```
library(metan)
cor_mat <- corr_coef(data_ge2, PH, EH, CD, CL, ED, NKR)
cor_mat$cor
reorder_cormat(cor_mat$cor)
```

---

resca *Rescale a variable to have specified minimum and maximum values*

Description

Helper function that rescales a continuous variable to have specified minimum and maximum values.

Usage

```
resca(
  .data = NULL,
  ...,
  values = NULL,
  new_min = 0,
  new_max = 100,
  keep = TRUE
)
```

Arguments

.data	The dataset. Grouped data is allowed.
...	Comma-separated list of unquoted variable names that will be rescaled.
values	Optional vector of values to rescale
new_min	The minimum value of the new scale. Default is 0.
new_max	The maximum value of the new scale. Default is 100
keep	Should all variables be kept after rescaling? If false, only rescaled variables will be kept.

Details

The function rescale a continuous variable as follows:

$$Rv_i = (Nmax - Nmin)/(Omax - Omin) * (O_i - Omax) + Nmax$$

Where Rv_i is the rescaled value of the ith position of the variable/ vector; $Nmax$ and $Nmin$ are the new maximum and minimum values; $OmaxandOmin$ are the maximum and minimum values of the original data, and O_i is the ith value of the original data.

There are basically two options to use `resca` to rescale a variable. The first is passing a data frame to `.data` argument and selecting one or more variables to be scaled using `....` The function will return the original variables in `.data` plus the rescaled variable(s) with the prefix `_res`. By using the function `group_by` from **dplyr** package it is possible to rescale the variable(s) within each level of the grouping factor. The second option is pass a numeric vector in the argument `values`. The output, of course, will be a numeric vector of rescaled values.

Value

A numeric vector if `values` is used as input data or a tibble if a data frame is used as input in `.data`.

Author(s)

Tiago Olivoto <tiagoolivoto@gmail.com>

Examples

```
library(metan)
library(dplyr)
# Rescale a numeric vector
resca(values = c(1:5))

 # Using a data frame
head(
 resca(data_ge, GY, HM, new_min = 0, new_max = 1)
)

# Rescale within factors;
# Select variables that stats with 'N' and ends with 'L';
# Compute the mean of these variables by ENV and GEN;
# Rescale the variables that ends with 'L' whithin ENV;
data_ge2 %>%
  select(ENV, GEN, starts_with("N"), ends_with("L")) %>%
  group_by(ENV, GEN) %>%
  summarise_all(mean) %>%
  group_by(ENV) %>%
  resca(ends_with("L")) %>%
  head(n = 13)
```

`Resende_indexes` *Stability indexes based on a mixed-effect model*

Description

This function computes the following indexes proposed by Resende (2007): the harmonic mean of genotypic values (HMGV), the relative performance of the genotypic values (RPGV) and the harmonic mean of the relative performance of genotypic values (HMRPGV).

Usage

```
Resende_indexes(.data)
```

Arguments

`.data`	An object of class `waasb`

Details

The indexes computed with this function have been used to select genotypes with stability performance in a mixed-effect model framework. Some examples are in Alves et al (2018), Azevedo Peixoto et al. (2018), Dias et al. (2018) and Colombari Filho et al. (2013).

The HMGV index is computed as

$$HMGV_i = \frac{1}{E}\sum_{j=1}^{E}\frac{1}{Gv_{ij}}$$

where E is the number of environments included in the analysis, Gv_{ij} is the genotypic value (BLUP) for the ith genotype in the jth environment.

The RPGV index is computed as

$$RPGV_i = \frac{1}{E}\sum_{j=1}^{E} BLUP_{ij}/\,\mu_j$$

The HMRPGV index is computed as

$$HMRPGV_i = \frac{1}{E}\sum_{j=1}^{E}\frac{1}{Gv_{ij}/\mu_j}$$

Value

A dataframe containing the indexes.

Author(s)

Tiago Olivoto <tiagoolivoto@gmail.com>

References

Alves, R.S., L. de Azevedo Peixoto, P.E. Teodoro, L.A. Silva, E.V. Rodrigues, M.D.V. de Resende, B.G. Laviola, and L.L. Bhering. 2018. Selection of Jatropha curcas families based on temporal stability and adaptability of genetic values. Ind. Crops Prod. 119:290-293. doi:10.1016/J.INDCROP.2018.04.029

Colombari Filho, J.M., M.D.V. de Resende, O.P. de Morais, A.P. de Castro, E.P. Guimaraes, J.A. Pereira, M.M. Utumi, and F. Breseghello. 2013. Upland rice breeding in Brazil: a simultaneous genotypic evaluation of stability, adaptability and grain yield. Euphytica 192:117-129. doi:10.1007/s10681-013-0922-2

Dias, P.C., A. Xavier, M.D.V. de Resende, M.H.P. Barbosa, F.A. Biernaski, R.A. Estopa. 2018. Genetic evaluation of Pinus taeda clones from somatic embryogenesis and their genotype x environment interaction. Crop Breed. Appl. Biotechnol. 18:55-64. doi:10.1590/1984-70332018v18n1a8

Azevedo Peixoto, L. de, P.E. Teodoro, L.A. Silva, E.V. Rodrigues, B.G. Laviola, and L.L. Bhering. 2018. Jatropha half-sib family selection with high adaptability and genotypic stability. PLoS One 13:e0199880. doi:10.1371/journal.pone.0199880

Resende MDV (2007) Matematica e estatistica na analise de experimentos e no melhoramento genetico. Embrapa Florestas, Colombo

Examples

```
library(metan)
res_ind <- waasb(data_ge,
                 env = ENV,
                 gen = GEN,
                 rep = REP,
                 resp = c(GY, HM))
model_indexes <- Resende_indexes(res_ind)

# Alternatively using the pipe operator %>%
res_ind <- data_ge %>%
           waasb(ENV, GEN, REP, c(GY, HM)) %>%
           Resende_indexes()
```

---

resp_surf *Response surface model*

Description

Compute a surface model and find the best combination of factor1 and factor2 to obtain the stationary point.

Usage

```
resp_surf(
  .data,
  factor1,
  factor2,
  rep = NULL,
  resp,
  prob = 0.05,
  verbose = TRUE
)
```

Arguments

.data	The dataset containing the columns related to Environments, factor1, factor2, replication/block and response variable(s).
factor1	The first factor, for example, dose of Nitrogen.
factor2	The second factor, for example, dose of potassium.
rep	The name of the column that contains the levels of the replications/blocks, if a designed experiment was conducted. Defaults to NULL.
resp	The response variable(s).
prob	The probability error.
verbose	If verbose = TRUE then some results are shown in the console.

Author(s)

Tiago Olivoto <tiagoolivoto@gmail.com>

Examples

```
library(metan)
# A small toy example

df = data.frame(
 expand.grid(x = seq(0, 4, by = 1),
             y = seq(0, 4, by = 1)),
 z = c(10, 11, 12, 11, 10,
       14, 15, 16, 15, 14,
       16, 17, 18, 17, 16,
       14, 15, 16, 15, 14,
       10, 11, 12, 11, 10)
)
mod = resp_surf(df, x, y, resp = z)
plot(mod)
```

`Schmildt` *Schmildt's genotypic confidence index*

Description

Stability analysis using the known genotypic confidence index (Annicchiarico, 1992) modified by Schmildt et al. 2011.

Usage

```
Schmildt(.data, env, gen, rep, resp, prob = 0.05, verbose = TRUE)
```

Arguments

`.data`	The dataset containing the columns related to Environments, Genotypes, replication/block and response variable(s)
`env`	The name of the column that contains the levels of the environments.
`gen`	The name of the column that contains the levels of the genotypes.
`rep`	The name of the column that contains the levels of the replications/blocks
`resp`	The response variable(s). To analyze multiple variables in a single procedure use, for example, `resp = c(var1,var2,var3)`.
`prob`	The probability of error assumed.
`verbose`	Logical argument. If `verbose = FALSE` the code will run silently.

Value

A list where each element is the result for one variable and contains the following data frames:

- **environments** Contains the mean, environmental index and classification as favorables and unfavorables environments.
- **general** Contains the genotypic confidence index considering all environments.
- **favorable** Contains the genotypic confidence index considering favorable environments.
- **unfavorable** Contains the genotypic confidence index considering unfavorable environments.

Author(s)

Tiago Olivoto, `<tiagoolivoto@gmail.com>`

References

Annicchiarico, P. 1992. Cultivar adaptation and recommendation from alfalfa trials in Northern Italy. J. Genet. Breed. 46:269-278.

Schmildt, E.R., A.L. Nascimento, C.D. Cruz, and J.A.R. Oliveira. 2011. Avaliacao de metodologias de adaptabilidade e estabilidade de cultivares milho. Acta Sci. - Agron. 33:51-58. doi:10.4025/actasciagron.v33i1.5817.

See Also

superiority,ecovalence,ge_stats,Annicchiarico

Examples

```
library(metan)
Sch <- Schmildt(data_ge2,
                env = ENV,
                gen = GEN,
                rep = REP,
                resp = PH)
print(Sch)
```

---

sem *Standard Error of the Mean*

Description

Helper function to compute the Standard Error of the Mean.

Usage

```
sem(x, ..., na.rm = TRUE)
```

Arguments

x	A numeric vector or a data frame.
...	Variables to compute the Standard Error of the Mean. If no variable is informed and x is a data frame, all the numeric variables will be used.
na.rm	A logical value indicating whether NA values should be stripped before computation proceeds.

Value

The Standard Error of the Mean(s) of x. If x is a numeric vector, the function returns a numeric value. If a data frame is used then a numeric vector with the Standard Error of the Mean for each numeric variable is returned.

See Also

gm_mean,hm_mean

Examples

```
num <- c(1:10, 50)
sem(num)

num_df <- make_mat(data_ge, ENV, GEN, GY)
sem(num_df)
```

`Shukla` *Shukla's stability variance parameter*

Description

The function computes the Shukla's stability variance parameter (1972) and uses the Kang's non-parametric stability (rank sum) to imcorporate the mean performance and stability into a single selection criteria.

Usage

```
Shukla(.data, env, gen, rep, resp, verbose = TRUE)
```

Arguments

`.data`	The dataset containing the columns related to Environments, Genotypes, replication/block and response variable(s).
`env`	The name of the column that contains the levels of the environments.
`gen`	The name of the column that contains the levels of the genotypes.
`rep`	The name of the column that contains the levels of the replications/blocks.
`resp`	The response variable(s). To analyze multiple variables in a single procedure use, for example, `resp = c(var1,var2,var3)`.
`verbose`	Logical argument. If `verbose = FALSE` the code will run silently.

Value

An object of class `Shukla`, which is a list containing the results for each variable used in the argument `resp`. For each variable, a tibble with the following columns is returned.

- **GEN** the genotype's code.
- **Y** the mean for the response variable.
- **ShuklaVar** The Shukla's stability variance parameter.
- **rMean** The rank for **Y** (decreasing).
- **rShukaVar** The rank for **ShukaVar**.
- **ssiShukaVar** The simultaneous selection index ($ssiShukaVar = rMean + rShukaVar$).

Author(s)

Tiago Olivoto <tiagoolivoto@gmail.com>

References

Shukla, G.K. 1972. Some statistical aspects of partitioning genotype-environmental components of variability. Heredity. 29:238-245. doi:10.1038/hdy.1972.87.

Kang, M.S., and H.N. Pham. 1991. Simultaneous Selection for High Yielding and Stable Crop Genotypes. Agron. J. 83:161. doi:10.2134/agronj1991.00021962008300010037x.

Examples

```
library(metan)
out <- Shukla(data_ge2,
              env = ENV,
              gen = GEN,
              rep = REP,
              resp = PH)
```

---

solve_svd *Pseudoinverse of a square matrix*

Description

This function computes the Moore-Penrose pseudoinverse of a square matrix using singular value decomposition.

Usage

```
solve_svd(x, tolerance = 2.220446e-16)
```

Arguments

x	A square matrix
tolerance	The tolerance to consider an eigenvalue equals to zero.

Value

A matrix with the same dimension of x.

Author(s)

Tiago Olivoto, <tiagoolivoto@gmail.com>

Examples

```
library(metan)
mat = matrix(c(1, 4, 2, 8), ncol = 2)
det(mat)
solve_svd(mat)
```

`split_factors` *Split a data frame by factors*

Description

Split a data frame into subsets grouping by one or more factors.

Usage

```
split_factors(.data, ..., keep_factors = FALSE, verbose = TRUE)
```

Arguments

`.data`	The data that will be split. Must contain at least one grouping variable.
`...`	Comma-separated list of unquoted variable names that will be used to group the data.
`keep_factors`	If more than two factors are in the dataframe, should the columns of the non-grouping factors be kept?
`verbose`	Logical argument. If `verbose = FALSE` the code will run silently.

Details

This function is used to split a data frame into a list where each element is a level of the grouping variable (or combination of grouping variables). By combining the function `split_factors` with the forward-pipe operator

element of the list.

Value

A list where each element is a named level of the grouping factors. If more than one grouping variable is used, then each element is the combination of the grouping variables.

Author(s)

Tiago Olivoto <tiagoolivoto@gmail.com>

Examples

```
library(metan)

g1 = split_factors(iris, Species)
g2 = split_factors(data_ge, ENV, keep_factors = TRUE)
```

`superiority` *Lin e Binns' superiority index*

Description

Nonparametric stability analysis using the superiority index proposed by Lin & Binns, (1992).

Usage

```
superiority(.data, env, gen, rep, resp, verbose = TRUE)
```

Arguments

`.data`	The dataset containing the columns related to Environments, Genotypes, replication/block and response variable(s)
`env`	The name of the column that contains the levels of the environments.
`gen`	The name of the column that contains the levels of the genotypes.
`rep`	The name of the column that contains the levels of the replications/blocks
`resp`	The response variable(s). To analyze multiple variables in a single procedure use, for example, `resp = c(var1,var2,var3)`.
`verbose`	Logical argument. If `verbose = FALSE` the code will run silently.

Value

An object of class `superiority` where each element is the result of one variable and contains the following items:

- **environments** The mean for each environment, the environment index and classification as favorable and unfavorable environments.
- **index** The superiority index computed for all, favorable and unfavorable environments.

Author(s)

Tiago Olivoto, `<tiagoolivoto@gmail.com>`

References

Lin, C.S., and M.R. Binns. 1988. A superiority measure of cultivar performance for cultivar x location data. Can. J. Plant Sci. 68:193-198. doi:10.4141/cjps88-018

See Also

`Annicchiarico`,`ecovalence`,`ge_stats`

Examples

```
library(metan)
out <- superiority(data_ge2,
                   env = ENV,
                   gen = GEN,
                   rep = REP,
                   resp = PH)
```

---

theme_metan *Personalized theme for ggplot2-based graphics*

Description

- `theme_metan()`: Theme with a gray background and major grids.
- `theme_metan_minimal()`: A minimalistic theme with half-open frame, white background, and no grid. For more details see `theme`.
- `transparent_color()`: A helper function to return a transparent color with Hex value of "#000000FF"

Usage

```
theme_metan(grid = "none", col.grid = "white", color.background = "gray95")

theme_metan_minimal()

transparent_color()
```

Arguments

`grid`	Control the grid lines in plot. Defaults to `"both"` (x and y major grids). Allows also `grid = "x"` for grids in x axis only, `grid = "y"` for grid in y axis only, or `grid = "none"` for no grids.
`col.grid`	The color for the grid lines
`color.background`	The color for the panel background.

Author(s)

Tiago Olivoto <tiagoolivoto@gmail.com>

`Thennarasu` *Thennarasu's stability statistics*

Description

Performs a stability analysis based on Thennarasu (1995) statistics.

Usage

```
Thennarasu(.data, env, gen, rep, resp, verbose = TRUE)
```

Arguments

`.data`	The dataset containing the columns related to Environments, Genotypes, replication/block and response variable(s).
`env`	The name of the column that contains the levels of the environments.
`gen`	The name of the column that contains the levels of the genotypes.
`rep`	The name of the column that contains the levels of the replications/blocks.
`resp`	The response variable(s). To analyze multiple variables in a single procedure use, for example, `resp = c(var1,var2,var3)`.
`verbose`	Logical argument. If `verbose = FALSE` the code will run silently.

Value

An object of class `Thennarasu`, which is a list containing the results for each variable used in the argument `resp`. For each variable, a tibble with the columns GEN, N1, N2, N3 and N4 is returned.

Author(s)

Tiago Olivoto <tiagoolivoto@gmail.com>

References

Thennarasu, K. 1995. On certain nonparametric procedures for studying genotype x environment interactions and yield stability. Ph.D. thesis. P.J. School, IARI, New Delhi, India.

Examples

```
library(metan)
out <- Thennarasu(data_ge, ENV, GEN, REP, GY)
```

to_factor *Encode variables to a factor*

Description

Function to quick mutate columns to factor.

Usage

```
to_factor(.data, ...)
```

Arguments

.data	A data frame
...	The variable(s) to encode to a factor.

Value

An object of the same class of .data with the variables in ... encoded to a factor.

Author(s)

Tiago Olivoto <tiagoolivoto@gmail.com>

Examples

```
library(metan)
PH_EH_to_factor <- to_factor(data_ge2, PH, EH)
PH_EH_to_factor <- to_factor(data_ge2, 4:5)
```

utils_num_str *Utilities for handling with numbers and strings*

Description

- all_lower_case(): Translate all non-numeric strings of a data frame to lower case.
- all_upper_case(): Translate all non-numeric strings of a data frame to upper case.
- extract_number(): Extract the number(s) of a string.
- extract_string(): Extract all strings, ignoring case.
- remove_strings(): Remove all strings of a variable.
- replace_number(): Replace numbers with a replacement.
- replace_string(): Replace all strings with a replacement, ignoring case.
- round_cols(): Round a selected column or a whole data frame to significant figures.

Usage

```
all_upper_case(.data)

all_lower_case(.data)

extract_number(
  .data,
  var,
  new_var = new_var,
  drop = FALSE,
  pull = FALSE,
  .before = NULL,
  .after = NULL
)

extract_string(
  .data,
  var,
  new_var = new_var,
  drop = FALSE,
  pull = FALSE,
  .before = NULL,
  .after = NULL
)

remove_strings(.data, ...)

replace_number(
  .data,
  var,
  new_var = new_var,
  pattern = NULL,
  replacement = "",
  drop = FALSE,
  pull = FALSE,
  .before = NULL,
  .after = NULL
)

replace_string(
  .data,
  var,
  new_var = new_var,
  pattern = NULL,
  replacement = "",
  drop = FALSE,
  pull = FALSE,
  .before = NULL,
```

```
  .after = NULL
)

round_cols(.data, ..., digits = 2)
```

Arguments

.data	A data frame
var	The variable to extract or replace numbers or strings.
new_var	The name of the new variable containing the numbers or strings extracted or replaced. Defaults to new_var.
drop	Logical argument. If TRUE keeps the new variable new_var and drops the existing ones. Defaults to FALSE.
pull	Logical argument. If TRUE, returns the last column (on the assumption that's the column you've created most recently), as a vector.
.before, .after	For replace_sting(), replace_number(), extract_string(), ,and extract_number() one-based column index or column name where to add the new columns.
...	The argument depends on the function used. • For round_cols() ... are the variables to round. If no variable is informed, all the numeric variables from data are used. • For remove_strings() ... are the variables to remove the strings. If no variable is informed, the strings of all non-numeric variables will be removed, keeping the numbers. If variables contains only strings, they will be replaced with NA.
pattern	A string to be matched. Regular Expression Syntax is also allowed.
replacement	A string for replacement.
digits	The number of significant figures.

Examples

```
library(metan)

################ Rounding numbers ###############
# All numeric columns
round_cols(data_ge2, digits = 1)

# Round specific columns
round_cols(data_ge2, EP, digits = 1)

########### Extract or replace numbers ##########
# Extract numbers
extract_number(data_ge, GEN)
extract_number(data_ge,
               var = GEN,
               drop = TRUE,
```

```
                new_var = g_number)

# Replace numbers

replace_number(data_ge, GEN)
replace_number(data_ge,
                var = GEN,
                pattern = "1",
                replacement = "_one",
                pull = TRUE)

########## Extract, replace or remove strings ##########
# Extract strings
extract_string(data_ge, GEN)
extract_string(data_ge,
                var = GEN,
                drop = TRUE,
                new_var = g_name)

# Replace strings
replace_string(data_ge, GEN)
replace_string(data_ge,
                var = GEN,
                new_var = GENOTYPE,
                pattern = "G",
                replacement = "GENOTYPE_")
# Remove strings
remove_strings(data_ge)
remove_strings(data_ge, ENV)
############# upper and lower cases ############
all_lower_case("GENOTYPE")
lc <- all_lower_case(data_ge)
lc
all_lower_case("GENOTYPE")

all_upper_case("Genotype")
all_upper_case(lc)
```

`utils_rows_cols` *Utilities for handling with rows and columns*

Description

- `add_cols()`: Add one or more columns to an existing data frame. If specified `.before` or `.after` columns does not exist, columns are appended at the end of the data. Return a data frame with all the original columns in `.data` plus the columns declared in `....` In `add_cols()` columns in `.data` are available for the expressions. So, it is possible to add a column based on existing data.

- `add_rows()`: Add one or more rows to an existing data frame. If specified `.before` or `.after` rows does not exist, rows are appended at the end of the data. Return a data frame with all the original rows in `.data` plus the rows declared in `....`
- `all_pairs()`: Get all the possible pairs between the levels of a factor.
- `column_exists()`: Checks if a column exists in a data frame. Return a logical value.
- `columns_to_first()`: Move columns to first positions in `.data`.
- `columns_to_last()`: Move columns to last positions in `.data`.
- `concatenate()`: Concatenate columns of a data frame. If `drop = TRUE` then the existing variables are dropped.
- `get_levels()`: Get the levels of a factor variable.
- `get_level_size()`: Get the size of each level of a factor variable.
- `remove_cols()`: Remove one or more columns from a data frame.
- `remove_rows()`: Remove one or more rows from a data frame.
- `reorder_cols()`: Reorder columns in a data frame.
- `select_cols()`: Select one or more columns from a data frame.
- `select_first_col()`: Select first variable, possibly with an offset.
- `select_last_col()`: Select last variable, possibly with an offset.
- `select_numeric_cols()`: Select all the numeric columns of a data frame.
- `select_non_numeric_cols()`: Select all the non-numeric columns of a data frame.
- `select_rows()`: Select one or more rows from a data frame.

Usage

```
add_cols(.data, ..., .before = NULL, .after = NULL)

add_rows(.data, ..., .before = NULL, .after = NULL)

all_pairs(.data, levels)

column_to_first(.data, ...)

column_to_last(.data, ...)

column_exists(.data, cols)

concatenate(
  .data,
  ...,
  new_var = new_var,
  sep = "_",
  drop = FALSE,
  pull = FALSE,
  .before = NULL,
  .after = NULL
```

```
)

get_levels(.data, group)

get_level_size(.data, group)

reorder_cols(.data, ..., .before = NULL, .after = NULL)

remove_cols(.data, ...)

remove_rows(.data, ...)

select_first_col(.data, offset = NULL)

select_last_col(.data, offset = NULL)

select_numeric_cols(.data)

select_non_numeric_cols(.data)

select_cols(.data, ...)

select_rows(.data, ...)
```

Arguments

.data	A data frame
...	The argument depends on the function used. • For add_cols() and add_rows() is name-value pairs. All values must have one element for each row in .data when using add_cols() or one element for each column in .data when using add_rows(). Values of length 1 will be recycled when using add_cols(). • For remove_cols() and select_cols(), ... is the column name or column index of the variable(s) to be dropped. • For columns_to_first() and columns_to_last(), ... is the column name or column index of the variable(s) to be moved to first or last in .data. • For remove_rows() and select_rows(), ... is an integer row value. • For concatenate(), ... is the unquoted variable names to be concatenated.
.before, .after	For add_cols(), concatenate(), and reorder_cols(), one-based column index or column name where to add the new columns, default: .after last column. For add_rows(), one-based row index where to add the new rows, default: .after last row.
levels	The levels of a factor or a numeric vector.
cols	A quoted variable name to check if it exists in .data.

new_var	The name of the new variable containing the concatenated values. Defaults to new_var.
sep	The separator to appear between concatenated variables. Defaults to "_".
drop	Logical argument. If TRUE keeps the new variable new_var and drops the existing ones. Defaults to FALSE.
pull	Logical argument. If TRUE, returns the last column (on the assumption that's the column you've created most recently), as a vector.
group	A factor variable to get the levels.
offset	Set it to *n* to select the *n*th variable from the end (for select_last_col()) of from the begin (for select_first_col())

Examples

```
library(metan)

################## Adding columns #################
# Variables x and y .after last column
data_ge %>%
  add_cols(x = 10,
           y = 30)
# Variables x and y .before the variable GEN
data_ge %>%
  add_cols(x = 10,
           y = 30,
           .before = "GEN")

# Creating a new variable based on the existing ones.
data_ge %>%
  add_cols(GY2 = GY^2,
           GY2_HM = GY2 + HM,
           .after = "GY")

############### Reordering columns ###############
reorder_cols(data_ge2, NKR, .before = "ENV")
reorder_cols(data_ge2, ENV, GEN, .after = "ED")

######## Selecting and removing columns ##########
select_cols(data_ge2, GEN, REP)
remove_cols(data_ge2, GEN, REP)

########## Selecting and removing rows ###########
select_rows(data_ge2, 2:3)
remove_rows(data_ge2, 2:3)

########### Concatenating columns ################
concatenate(data_ge, ENV, GEN, REP)
concatenate(data_ge, ENV, GEN, REP, drop = TRUE)

# Combine with add_cols() and replace_string()
```

```
data_ge2 %>%
 add_cols(ENV_GEN = concatenate(., ENV, GEN, pull = TRUE),
          .after = "GEN") %>%
 replace_string(ENV_GEN,
                pattern = "H",
                replacement = "HYB_",
                .after = "ENV_GEN")


################### Adding rows ##################
data_ge %>%
  add_rows(ENV = "E_TEST",
           GEN = "G_TEST",
           REP = 3,
           GY = 10.3,
           HM = 100.11,
           .after = 1)
########## checking if a column exists ###########
column_exists(data_g, "GEN")

####### get the levels and size of levels ########
get_levels(data_g, GEN)
get_level_size(data_g, GEN)

############## all possible pairs ################
all_pairs(data_g, GEN)

########## select numeric variables only #########
select_numeric_cols(data_g)
select_non_numeric_cols(data_g)
```

---

waas *Weighted Average of Absolute Scores*

Description

Compute the Weighted Average of Absolute Scores for AMMI analysis (Olivoto et al., 2019).

Usage

```
waas(
  .data,
  env,
  gen,
  rep,
  resp,
  mresp = NULL,
  wresp = NULL,
```

```
  prob = 0.05,
  naxis = NULL,
  ind_anova = TRUE,
  verbose = TRUE
)
```

Arguments

.data	The dataset containing the columns related to Environments, Genotypes, replication/block and response variable(s).
env	The name of the column that contains the levels of the environments.
gen	The name of the column that contains the levels of the genotypes.
rep	The name of the column that contains the levels of the replications/blocks.
resp	The response variable(s). To analyze multiple variables in a single procedure a vector of variables may be used. For example resp = c(var1,var2,var3).
mresp	A numeric vector of the same length of resp. The mresp will be the new maximum value after rescaling. By default, all variables in resp are rescaled so that de maximum value is 100 and the minimum value is 0.
wresp	The weight for the response variable(s) for computing the WAASBY index. Must be a numeric vector of the same length of resp. Defaults to 50, i.e., equal weights for stability and mean performance.
prob	The p-value for considering an interaction principal component axis significant.
naxis	The number of IPCAs to be used for computing the WAAS index. Default is NULL (Significant IPCAs are used). If values are informed, the number of IPCAS will be used independently on its significance. Note that if two or more variables are included in resp, then naxis must be a vector.
ind_anova	Logical argument set to TRUE. If FALSE the within-environment ANOVA is not performed.
verbose	Logical argument. If verbose = FALSE the code is run silently.

Details

This function compute the weighted average of absolute scores, estimated as follows:

$$WAAS_i = \sum_{k=1}^{p} |IPCA_{ik} \times EP_k| / \sum_{k=1}^{p} EP_k$$

where $WAAS_i$ is the weighted average of absolute scores of the *i*th genotype; PCA_{ik} is the score of the *i*th genotype in the *k*th IPCA; and EP_k is the explained variance of the *k*th IPCA for *k = 1,2,..,p*, considering *p* the number of significant PCAs, or a declared number of PCAs. For example if prob = 0.05, all axis that are significant considering this probability level are used. The number of axis can be also informed by declaring naxis = x. This will override the number of significant axes according to the argument codeprob.

Value

An object of class `waas` with the following items for each variable:

- **individual** A within-environments ANOVA considering a fixed-effect model.
- **model** A data frame with the response variable, the scores of all Principal Components, the estimates of Weighted Average of Absolute Scores, and WAASY (the index that consider the weights for stability and productivity in the genotype ranking.
- **MeansGxE** The means of genotypes in the environments
- **PCA** Principal Component Analysis.
- **anova** Joint analysis of variance for the main effects and Principal Component analysis of the interaction effect.
- **Details** A list summarizing the results. The following information are showed. `WgtResponse`, the weight for the response variable in estimating WAASB, `WgtWAAS` the weight for stability, `Ngen` the number of genotypes, `Nenv` the number of environments, `OVmean` the overall mean, `Min` the minimum observed (returning the genotype and environment), `Max` the maximum observed, `Max` the maximum observed, `MinENV` the environment with the lower mean, `MaxENV` the environment with the larger mean observed, `MinGEN` the genotype with the lower mean, `MaxGEN` the genotype with the larger.
- **residuals** The residuals of the model.
- **probint** The p-value for the genotype-vs-environment interaction.

Author(s)

Tiago Olivoto <tiagoolivoto@gmail.com>

References

Olivoto, T., A.D.C. L\'ucio, J.A.G. da silva, V.S. Marchioro, V.Q. de Souza, and E. Jost. 2019a. Mean performance and stability in multi-environment trials I: Combining features of AMMI and BLUP techniques. Agron. J. 111:2949-2960. doi:10.2134/agronj2019.03.0220

See Also

`waasb`

Examples

```
library(metan)

# Considering p-value <= 0.05 to compute the WAAS

model <- waas(data_ge,
              env = ENV,
              gen = GEN,
              rep = REP,
              resp = GY)
```

```
# Declaring the number of axis to be used for computing WAAS
# and assigning a larger weight for the response variable when
# computing the WAASBY index.

model2 <- waas(data_ge,
               env = ENV,
               gen = GEN,
               rep = REP,
               resp = GY,
               naxis = 3,
               wresp = 60)

# Analyzing multiple variables (GY and HM) at the same time
# considering that smaller values of HM are better and higher
# values of GY are better, assigning a larger weight for the GY
# and a smaller weight for HM when computing WAASBY index.

model3 <- waas(data_ge,
               env = ENV,
               gen = GEN,
               rep = REP,
               resp = c(GY, HM),
               mresp = c(100, 0),
               wresp = c(60, 40))
```

---

waasb *Weighted Average of Absolute Scores*

Description

Compute the Weighted Average of Absolute Scores for quantifying the stability in multi-environment trials using mixed-effect models (Olivoto et al., 2019).

Usage

```
waasb(
  .data,
  env,
  gen,
  rep,
  resp,
  mresp = NULL,
  wresp = NULL,
  random = "gen",
  prob = 0.05,
  ind_anova = TRUE,
```

```
  verbose = TRUE
)
```

Arguments

.data	The dataset containing the columns related to Environments, Genotypes, replication/block and response variable(s).
env	The name of the column that contains the levels of the environments.
gen	The name of the column that contains the levels of the genotypes.
rep	The name of the column that contains the levels of the replications/blocks.
resp	The response variable(s). To analyze multiple variables in a single procedure a vector of variables may be used. For example resp = c(var1,var2,var3).
mresp	A numeric vector of the same length of resp. The mresp will be the new maximum value after rescaling. By default, all variables in resp are rescaled so that de maximum value is 100 and the minimum value is 0.
wresp	The weight for the response variable(s) for computing the WAASBY index. Must be a numeric vector of the same length of resp. Defaults to 50, i.e., equal weights for stability and mean performance.
random	The effects of the model assumed to be random. Default is random = "gen" (genotype and genotype-vs-environment as random effects. Other values allowed are random = "env" (environment, genotype-vs-environment and block-within-environment random effects) or random = "all" all effects except the intercept are assumed to be random effects.
prob	The probability for estimating confidence interval for BLUP's prediction.
ind_anova	Logical argument set to TRUE. If FALSE the within-environment ANOVA is not performed.
verbose	Logical argument. If verbose = FALSE the code are run silently.

Details

This function compute the weighted average of absolute scores considering all principal component axis from the Singular Value Decomposition (SVD) of the BLUP'S GxE effects matrix generated by a linear mixed-effect model.

Value

An object of class waasb with the following items for each variable: * **individual** A within-environments ANOVA considering a fixed-effect model.

- **fixed** Test for fixed effects.
- **random** Variance components for random effects.
- **LRT** The Likelihood Ratio Test for the random effects.
- **model** A data frame with the response variable, the scores of all Principal Components, the estimates of Weighted Average of Absolute Scores, and WAASY (the index that consider the weights for stability and productivity in the genotype ranking.

- **blupGEN** The estimated BLUPS for genotypes (If `random = "gen"` or `random = "all"`)
- **BLUPenv** The estimated BLUPS for environments, (If `random = "env"` or `random = "all"`).
- **BLUPge** The estimated BLUPS of all genotypes in all environments "BLUPij".
- **PCA** The results of Principal Component Analysis with eigenvalues and explained variance of BLUP-interaction matrix.
- **MeansGxE** The phenotypic means of genotypes in the environments, with observed, predicted and OLS residual prediction.
- **Details** A list summarizing the results. The following information are showed. `WgtResponse`, the weight for the response variable in estimating WAASB, `WgtWAAS` the weight for stability, `Ngen` the number of genotypes, `Nenv` the number of environments, `OVmean` the overall mean, `Min` the minimum observed (returning the genotype and environment), `Max` the maximum observed, `Max` the maximum observed, `MinENV` the environment with the lower mean, `MaxENV` the environment with the larger mean observed, `MinGEN` the genotype with the lower mean, `MaxGEN` the genotype with the larger.
- **ESTIMATES** A list with the following values: `GEV` the genotype-by-environment variance (and percentage of phenotypic variance); `GV` the genotypic variance (and percentage of phenotypic variance); `EV` the environmental variance;`RV` the residual variance (and percentage of phenotypic variance); `FV` the phenotypic variance; `h2g` the heritability of the trait; `GEr2` the coefficient of determination of the interaction effects; `h2mg` the heritability of the mean; `AccuGen` the selective accuracy; `rge` the genotype-environment correlation; `CVg` the genotypic coefficient of variation; `CVr` the residual coefficient of variation; `CVratio` the ratio between genotypic and residual coefficient of variation * **residuals** The residuals of the model.

Author(s)

Tiago Olivoto <tiagoolivoto@gmail.com>

References

Olivoto, T., A.D.C. L\'ucio, J.A.G. da silva, V.S. Marchioro, V.Q. de Souza, and E. Jost. 2019. Mean performance and stability in multi-environment trials I: Combining features of AMMI and BLUP techniques. Agron. J. 111:2949-2960. doi:10.2134/agronj2019.03.0220

See Also

`waas`

Examples

```
library(metan)

# Genotypes as random effects
# Equal weights for response variable and stability

model <- waasb(data_ge,
               env = ENV,
               gen = GEN,
```

```
              rep = REP,
              resp = GY)

# Higher weight for response variable

model2 <- waasb(data_ge,
               env = ENV,
               gen = GEN,
               rep = REP,
               resp = GY,
               wresp = 65)

# Environment as random effects analyzing more than one variable
# Smaller values of HM are better and higher
# Higher values of GY are better
# Larger weight for the GY (60%)
# Smaller weight for HM (40%)

model3 <- waasb(data_ge,
                env = ENV,
                gen = GEN,
                rep = REP,
                resp = c(GY, HM),
                random = "env",
                mresp = c(100, 0),
                wresp = c(60, 40))
```

`waas_means` *Weighted Average of Absolute Scores*

Description

Compute the Weighted Average of Absolute Scores (Olivoto et al., 2019) based on means for genotype-environment data as follows:

Usage

```
waas_means(
  .data,
  env,
  gen,
  resp,
  mresp = NULL,
  wresp = NULL,
  min_expl_var = 85,
  verbose = TRUE
)
```

Arguments

`.data`	The dataset containing the columns related to Environments, Genotypes, replication/block and response variable(s).
`env`	The name of the column that contains the levels of the environments.
`gen`	The name of the column that contains the levels of the genotypes.
`resp`	The response variable(s). To analyze multiple variables in a single procedure a vector of variables may be used. For example `resp = c(var1,var2,var3)`. Select helpers are also allowed.
`mresp`	A numeric vector of the same length of `resp`. The `mresp` will be the new maximum value after rescaling. By default, all variables in `resp` are rescaled so that de maximum value is 100 and the minimum value is 0.
`wresp`	The weight for the response variable(s) for computing the WAASBY index. Must be a numeric vector of the same length of `resp`. Defaults to 50, i.e., equal weights for stability and mean performance.
`min_expl_var`	The minimum explained variance. Defaults to 85. Interaction Principal Compoment Axis are iteractively retained up to the explained variance (eigenvalues in the singular value decomposition of the matrix with the interaction effects) be greather than or equal to `min_expl_var`. For example, if the explained variance (in percentage) in seven possible IPCAs are `56,21,9,6,4,3,1`, resulting in a cumulative proportion of `56,77,86,92,96,99,100`, then `p = 3`, i.e., three IPCAs will be used to compute the index WAAS.
`verbose`	Logical argument. If `verbose = FALSE` the code is run silently.

Details

$$WAAS_i = \sum_{k=1}^{p} |IPCA_{ik} \times EP_k| / \sum_{k=1}^{p} EP_k$$

where $WAAS_i$ is the weighted average of absolute scores of the *i*th genotype; PCA_{ik} is the score of the *i*th genotype in the *k*th IPCA; and EP_k is the explained variance of the *k*th IPCA for *k = 1,2,..,p*, where *p* is the number of IPCAs that explain at least an amount of the genotype-interaction variance declared in the argument `min_expl_var`.

Value

An object of class `waas_means` with the following items for each variable:

- **model** A data frame with the response variable, the scores of all Principal Components, the estimates of Weighted Average of Absolute Scores, and WAASY (the index that consider the weights for stability and productivity in the genotype ranking.
- **ge_eff** A *gxe* matrix containing the genotype-environment effects.
- **eigenvalues** The eigenvalues from the singular value decomposition of the matrix withe the genotype-environment interaction effects.
- **proportion** The proportion of the variance explained by each IPCA.
- **cum_proportion** The cumulative proportion of the variance explained.

Author(s)

Tiago Olivoto <tiagoolivoto@gmail.com>

References

Olivoto, T., A.D.C. L\'ucio, J.A.G. da silva, V.S. Marchioro, V.Q. de Souza, and E. Jost. 2019a. Mean performance and stability in multi-environment trials I: Combining features of AMMI and BLUP techniques. Agron. J. 111:2949-2960. doi:10.2134/agronj2019.03.0220

See Also

waas waasb

Examples

```
library(metan)
# Data with replicates
model <- waas(data_ge,
              env = ENV,
              gen = GEN,
              rep = REP,
              resp = everything())

# Based on means of genotype-environment data
data_means <- means_by(data_ge, ENV, GEN)
model2 <- waas_means(data_ge,
                     env = ENV,
                     gen = GEN,
                     resp = everything())
# The index WAAS
get_model_data(model, what = "OrWAASB")
get_model_data(model2, what = "OrWAASB")
```

---

wsmp *Weighting between stability and mean performance*

Description

This function computes the WAASY or WAASBY indexes (Olivoto et al., 2019) considering different scenarios of weights for stability and mean performance.

Usage

```
wsmp(
  model,
  mresp = 100,
  increment = 5,
  saveWAASY = 50,
  prob = 0.05,
  progbar = TRUE
)
```

Arguments

model	Should be an object of class waas or waasb.
mresp	A numeric value that will be the new maximum value after rescaling. By default, the variable in resp is rescaled so that the original maximum and minimum values are 100 and 0, respectively. Let us consider that for a specific trait, say, lodging incidence, lower values are better. In this case, you should use mresp = 0 to rescale the response variable so that the lowest values will become 100 and the highest values 0.
increment	The increment in the weight ratio for stability and mean performance. Se the **Details** section for more information.
saveWAASY	Automatically save the WAASY values when the weight for stability is saveWAASY. Default is 50. Please, note that saveWAASY
prob	The p-value for considering an interaction principal component axis significant. must be multiple of increment. If this assumption is not valid, an error will be occur.
progbar	A logical argument to define if a progress bar is shown. Default is TRUE.

Details

After fitting a model with the functions waas or waasb it is possible to compute the superiority indexes WAASY or WAASBY in different scenarios of weights for stability and mean performance. The number of scenarios is defined by the arguments increment. By default, twenty-one different scenarios are computed. In this case, the the superiority index is computed considering the following weights: stability (waasb or waas) = 100; mean performance = 0. In other words, only stability is considered for genotype ranking. In the next iteration, the weights becomes 95/5 (since increment = 5). In the third scenario, the weights become 90/10, and so on up to these weights become 0/100. In the last iteration, the genotype ranking for WAASY or WAASBY matches perfectly with the ranks of the response variable.

Value

An object of class wsmp with the following items for each variable:

- **scenarios** A list with the model for all computed scenarios.
- **WAASY** The values of the WAASY estimated when the weight for the stability in the loop match with argument saveWAASY.

- **hetdata, hetcomb** The data used to produce the heatmaps.
- **Ranks** All the values of WAASY estimated in the different scenarios of WAAS/GY weighting ratio.

Author(s)

Tiago Olivoto <tiagoolivoto@gmail.com>

References

Olivoto, T., A.D.C. L\'ucio, J.A.G. da silva, V.S. Marchioro, V.Q. de Souza, and E. Jost. 2019. Mean performance and stability in multi-environment trials I: Combining features of AMMI and BLUP techniques. Agron. J. doi:10.2134/agronj2019.03.0220

See Also

resca

Examples

```
library(metan)
model <- waas(data_ge2,
              env = ENV,
              gen = GEN,
              rep = REP,
              resp = PH)
scenarios <- wsmp(model)
```

Index