Package ‘metan’

April 25, 2020

Type Package
Title Multi Environment Trials Analysis
Version 1.5.1
Maintainer Tiago Olivoto <tiagoolivoto@gmail.com>

License GPL-3

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

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

Depends R (>= 3.5.0)
R topics documented:

**Imports** ade4, cowplot, dplyr, FWDselect, GGally, ggforce, ggplot2 (>= 3.3.0), ggrepel, grid, lme4, lmerTest, magrittr, methods, progress, purrr, rlang, tibble, tidyr, tidyselect (>= 1.0.0)

**Suggests** DT, knitr, readxl, rmarkdown, roxygen2

**VignetteBuilder** knitr

**Encoding** UTF-8

**Language** en-US

**LazyData** true

**RoxygenNote** 7.1.0

**NeedsCompilation** no

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

**Repository** CRAN

**Date/Publication** 2020-04-25 14:40:06 UTC

R topics documented:

```
metan-package .................................................................................. 5
AMMI_indexes .................................................................................. 6
Annicchiarico .................................................................................. 8
anova_ind .................................................................................... 9
anova_joint .................................................................................. 11
arrange_ggplot ............................................................................. 13
as.lpcor ....................................................................................... 14
barplots ..................................................................................... 15
bind_cv ....................................................................................... 19
can_corr ..................................................................................... 20
clustering ................................................................................... 22
colindiag .................................................................................... 24
comb_vars .................................................................................. 26
corr_ci ....................................................................................... 27
corr_coef .................................................................................... 28
corr_plot ..................................................................................... 29
corr_ss ....................................................................................... 32
corr_stab_ind ............................................................................. 34
covecor_design ........................................................................... 35
cv_ammi ...................................................................................... 36
cv_ammif ................................................................................... 39
cv_blup ....................................................................................... 41
data_alpha .................................................................................. 43
data_g ......................................................................................... 44
data_ge ....................................................................................... 45
data_ge2 ..................................................................................... 46
desc_stat ..................................................................................... 47
doo ............................................................................................. 50
```
R topics documented:

ecovalence ................................................................. 51
env_dissimilarity ....................................................... 52
fai_blup ................................................................. 53
find_outliers ............................................................ 54
Fox ......................................................................... 55
gafem ................................................................. 57
gai ................................................................. 59
gamem ................................................................. 60
gamem_met .............................................................. 62
get_model_data ........................................................ 66
ge_cluster ............................................................... 72
df_details ............................................................... 74
df_effects ............................................................... 75
df_factanal ............................................................. 76
df_means ............................................................... 78
df_plot ............................................................... 79
df_reg ............................................................... 80
df_stats ............................................................... 81
df_winners ......................................................... 83
gge ................................................................. 85
gtb ................................................................. 87
Huehn ............................................................... 89
impute_missing_val .................................................. 90
inspect .............................................................. 92
int.effects ............................................................ 93
is.lpcor .............................................................. 94
is_balanced_trial ..................................................... 94
lineplots ............................................................. 95
lpcor .............................................................. 98
mahala ............................................................. 99
mahala_design ........................................................ 100
make_long .......................................................... 101
make_mat ........................................................... 102
meansGxE ............................................................. 103
mgidi .............................................................. 104
mtsi .............................................................. 105
non_collinear_vars ................................................... 107
pairs_mantel .......................................................... 108
path_coeff .......................................................... 111
performs_ammi .................................................... 113
plot.anova_joint .................................................... 115
plot.can_cor ........................................................ 116
plot.clustering .................................................... 118
plot.corr_coef ..................................................... 119
plot.cvalidation ................................................... 121
plot.env_dissimilarity ........................................... 123
plot.fai_blup ........................................................ 124
plot.gafem ........................................................ 125
<table>
<thead>
<tr>
<th>R topics documented:</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>plot.gamem</td>
<td>126</td>
</tr>
<tr>
<td>plot.ge_cluster</td>
<td>128</td>
</tr>
<tr>
<td>plot.ge_effects</td>
<td>129</td>
</tr>
<tr>
<td>plot.ge_factanal</td>
<td>130</td>
</tr>
<tr>
<td>plot.ge_reg</td>
<td>132</td>
</tr>
<tr>
<td>plot.gge</td>
<td>134</td>
</tr>
<tr>
<td>plot.mgidi</td>
<td>137</td>
</tr>
<tr>
<td>plot.mtsi</td>
<td>138</td>
</tr>
<tr>
<td>plot_performs_ammi</td>
<td>139</td>
</tr>
<tr>
<td>plot.resp_surf</td>
<td>140</td>
</tr>
<tr>
<td>plot.waas</td>
<td>141</td>
</tr>
<tr>
<td>plot.waasb</td>
<td>142</td>
</tr>
<tr>
<td>plot.wsmp</td>
<td>144</td>
</tr>
<tr>
<td>plot_blup</td>
<td>145</td>
</tr>
<tr>
<td>plot_ci</td>
<td>147</td>
</tr>
<tr>
<td>plot_eigen</td>
<td>149</td>
</tr>
<tr>
<td>plot_scores</td>
<td>150</td>
</tr>
<tr>
<td>plot_waasby</td>
<td>155</td>
</tr>
<tr>
<td>predict.gamem</td>
<td>157</td>
</tr>
<tr>
<td>predict.gge</td>
<td>157</td>
</tr>
<tr>
<td>predict_performs_ammi</td>
<td>159</td>
</tr>
<tr>
<td>predict.waas</td>
<td>160</td>
</tr>
<tr>
<td>predict.waasb</td>
<td>161</td>
</tr>
<tr>
<td>print.AMMI_indexes</td>
<td>162</td>
</tr>
<tr>
<td>print.Annicchiarico</td>
<td>163</td>
</tr>
<tr>
<td>print.anova_ind</td>
<td>164</td>
</tr>
<tr>
<td>print.anova_joint</td>
<td>164</td>
</tr>
<tr>
<td>print.can_cor</td>
<td>165</td>
</tr>
<tr>
<td>print.colindia</td>
<td>166</td>
</tr>
<tr>
<td>print.corr_coef</td>
<td>167</td>
</tr>
<tr>
<td>print.ecovalence</td>
<td>167</td>
</tr>
<tr>
<td>print.env_dissimilarity</td>
<td>168</td>
</tr>
<tr>
<td>print.Fox</td>
<td>169</td>
</tr>
<tr>
<td>print.gamem</td>
<td>170</td>
</tr>
<tr>
<td>print.ge_factanal</td>
<td>171</td>
</tr>
<tr>
<td>print.ge_reg</td>
<td>172</td>
</tr>
<tr>
<td>print_ge_stats</td>
<td>172</td>
</tr>
<tr>
<td>print.Huehn</td>
<td>173</td>
</tr>
<tr>
<td>print.lpcor</td>
<td>174</td>
</tr>
<tr>
<td>print.mgidi</td>
<td>175</td>
</tr>
<tr>
<td>print.mtsi</td>
<td>176</td>
</tr>
<tr>
<td>print.path_coeff</td>
<td>177</td>
</tr>
<tr>
<td>print_performs_ammi</td>
<td>178</td>
</tr>
<tr>
<td>print.Schmildt</td>
<td>178</td>
</tr>
<tr>
<td>print.Shukla</td>
<td>179</td>
</tr>
<tr>
<td>print_superiority</td>
<td>180</td>
</tr>
<tr>
<td>print.Thennarasu</td>
<td>181</td>
</tr>
<tr>
<td>print.waas</td>
<td>182</td>
</tr>
</tbody>
</table>
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:

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

where \( r \) is the number of replications included in the analysis.

The SIPC index is computed as follows:

$$SIPC_i = \sum_{k=1}^{r} |\lambda_{0.5}^{0.5} a_{ik}|$$
where \( P \) is the number of IPCA retained via F-tests. The EV index is computed as follows:

\[
EV_i = \sum_{k=1}^{P} \frac{a_{ik}^2}{P}
\]

The ZA index is computed as follows:

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

where \( \theta_k \) is the percentage sum of squares explained by the \( k \)th IPCA.

\[
WAAS_i = \frac{\sum_{k=1}^{P} |IPCA_{ik} \times EP_k|}{\sum_{k=1}^{P} EP_k}
\]

where \( WAAS_i \) is the weighted average of absolute scores of the \( i \)th genotype; \( IPCA_{ik} \) 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**


Annicchiarico

Examples

```r
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

```r
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.
analyses.

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 favorable and unfavorable 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

See Also
superiority, ecovalence, ge_stats

Examples

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

---

**anova_ind**

*Within-environment analysis of variance*

**Description**
Performs a within-environment analysis of variance in randomized complete block or alpha-lattice designs and returns values such as Mean Squares, p-values, coefficient of variation, heritability, and accuracy of selection.

**Usage**

```r
anova_ind(.data, env, gen, rep, resp, block = NULL)
```
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)`.

.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, except the error, are assumed to be fixed.**

Value

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

1. **individual**: A tidy tbl_df with the results of the individual analysis of variance with the following column names:

   - **For analysis in alpha-lattice designs**: ENV: The environment code; MEAN: The grand mean; MSG, MSCR, MSIB_R: The mean squares for genotype, replicates and incomplete blocks within replicates, respectively. FCG, FCR, FCIB_R: The F-calculated for genotype, replicates and incomplete blocks within replicates, respectively. PFG, PFCR, PFIB_R: The P-values for genotype, replicates and incomplete blocks within replicates, respectively. MSE: The mean square error. CV: coefficient of variation. h2: broad-sense heritability. AS: accuracy of selection (square root of h2)

   - **For analysis in randomized complete block design**: MSG, MSB: The mean squares for genotype and blocks, respectively. FCG, FCB: The F-calculated for genotype and blocks, respectively. PFG, PFB: The P-values for genotype and blocks, respectively. MSE: The mean square error. CV: coefficient of variation. h2: broad-sense heritability. AS: accuracy of selection (square root of h2)

1. **MSRatio** The ratio between the higher and lower residual mean square.

Author(s)

Tiago Olivoto <tiagoolivoto@gmail.com>

References

**Examples**

```r
library(metan)
# ANOVA for all variables in data
ind_an <- anova_ind(data_ge,
  env = ENV,
  gen = GEN,
  rep = REP,
  resp = everything())
# mean for each environment
get_model_data(ind_an)

# P-value for genotype effect
get_model_data(ind_an, "PFG")
```

---

**anova_joint**  
*Joint analysis of variance*

**Description**

Performs a joint analysis of variance to check for the presence of genotype-vs-environment interactions using both randomized complete block and alpha-lattice designs.

**Usage**

```r
anova_joint(.data, env, gen, rep, resp, block = NULL, 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)`.

- `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, except the error, are assumed to be fixed.**

- `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 following objects:

- **anova**: The two-way ANOVA table
- **model**: The model of class `lm`.
- **augment**: Information about each observation in the dataset. This includes predicted values in the 'fitted' column, residuals in the 'resid' column, standardized residuals in the 'stdres' column, the diagonal of the 'hat' matrix in the 'hat' column, and standard errors for the fitted values in the 'se.fit' column.
- **details**: A tibble with the following data: \( N_{gen} \), the number of genotypes; \( OV_{mean} \), the grand mean; \( Min \), the minimum observed (returning the genotype and replication/block); \( Max \) the maximum observed, \( Min_{GEN} \) the loser winner genotype, \( Max_{GEN} \), the winner genotype.

Author(s)

Tiago Olivoto <tiagoolivoto@gmail.com>

References


See Also

`get_model_data` `anova_ind`

Examples

```r
library(metan)
# traditional usage approach
j_an <- anova_joint(data_ge,
                     env = ENV,
                     gen = GEN,
                     rep = REP,
                     resp = everything())

# Predicted values
get_model_data(j_an)

# Details
get_model_data(j_an, "details")
```
arrange_ggplot

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

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

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.
- `align`: Specifies whether graphs in the grid should be horizontally ("h") or vertically ("v") aligned. "hv" (default) align in both directions, "none" do not align the plot.

Value

None.

Examples

```r
library(ggplot2)
library(metan)
pl <- ggplot(mtcars, aes(wt, mpg)) +
```
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)
barplots

Fast way to create bar plots

Description

- `plot_bars()` Creates a bar plot based on one categorical variable and one numeric variable. It can be used to show the results of a one-way trial with qualitative treatments.
- `plot_factbars()` Creates a bar plot based on two categorical variables and one numeric variable. It can be used to show the results of a two-way trial with qualitative-qualitative treatment structure.

Usage

```r
plot_bars(
  .data,  
  x, 
  y, 
  order = NULL, 
  y.lim = NULL, 
  y.breaks = waiver(), 
  y.expand = 0.05, 
  xlab = NULL, 
  ylab = NULL, 
  n.dodge = 1, 
  check.overlap = FALSE, 
  color.bar = "black", 
  fill.bar = "gray", 
  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 = NULL, 
  level = 0.95, 
  invert = FALSE, 
  width.bar = 0.9, 
  size.line = 0.5, 
  size.text = 12, 
  fontfam = "sans", 
  na.rm = TRUE, 
  verbose = FALSE,
```

plot_theme = theme_metan()
)

plot_factbars(
  .data,
  ..., 
  resp,
  y.lim = NULL,
  y.breaks = waiver(),
  y.expand = 0.05,
  xlab = NULL,
  ylab = NULL,
  n.dodge = 1,
  check.overlap = FALSE,
  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 = NULL,
  level = 0.95,
  invert = FALSE,
  col = TRUE,
  palette = "Spectral",
  width.bar = 0.9,
  legend.position = "bottom",
  size.line = 0.5,
  size.text = 12,
  fontfam = "sans",
  na.rm = TRUE,
  verbose = FALSE,
  plot_theme = theme_metan()
)

**Arguments**

- `.data` The data set.
- `x, y` Argument valid for `plot_bars()`. The variables to be mapped to the x and y axes, respectively.
- `order` Argument valid for `plot_bars()`. Controls the order of the factor in the x axis. Defaults to the order of the factors in `.data`. Use `order = "asce"` or `order = "desc"` to reorder the labels to ascending or descending order, respectively, based on the values of the variable `y`.
The range of y axis. Defaults to NULL (maximum and minimum values of the
data set). New values can be inserted as y.lim = c(y.min, y.max).

The breaks to be plotted in the y-axis. Defaults to waiver(). authomatic
breaks. The same arguments than x.breaks can be used.

A multiplication range expansion factor. Defaults to 0.05.

The labels of the axes x and y, respectively. Defaults to NULL.

The number of rows that should be used to render the x labels. This is useful for
displaying labels that would otherwise overlap.

Silently remove overlapping labels, (recursively) prioritizing the first, last, and
middle labels.

Argument valid for plot_bars(). The color and fill values of the bars.

A vector of characters to show in each bar. Defaults to NULL.

The horizontal and vertical adjust for the labels in the bar. Defaults to 0.5 and
-0.5, respectively.

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.

The size of the text in the bar labels.

The horizontal and vertical adjust for the labels in the bar. Defaults to 0.5 and 1,
respectively.

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.

Logical argument, set to TRUE. In this case, an error bar is shown.

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.

The width of the error bar. Defaults to 30% of width.bar.

The confidence level

Logical argument. If TRUE, rotate the plot in plot_bars() and invert the order
of the factors in plot_factbars().

The width of the bars in the graph. Defaults to 0.9. Possible values are in the
range 0-1.

The size of the line in the bars. Default to 0.5.

The size of the text. Default to 12.

The family of the font text. Defaults to "sans".

Should 'NA' values be removed to compute the statistics? Defaults to true

Logical argument. If TRUE a tibble containing the mean, N, standard deviation,
standard error of mean and confidence interval is returned.
plot_theme
The graphical theme of the plot. Default is plot_theme = theme_metan(). For more details, see theme.

... Argument valid for plot_factbars(). A comma-separated list of unquoted variable names. Sets the two variables to be mapped to the x axis.

resp Argument valid for plot_factbars(). The response variable to be mapped to the y axis.

col Logical argument valid for plot_factbars(). If FALSE, a gray scale is used.

palette Argument valid for plot_factbars(). The color palette to be used. For more details, see ?scale_colour_brewer

legend.position The position of the legend in the plot.

Value
An object of class gg, ggplot.

Author(s)
Tiago Olivoto <tiagoolivoto@gmail.com>

See Also
plot_lines, plot_factlines

Examples

library(metan)
# two categorical variables
plot_factbars(data_ge2,
   GEN,
   ENV,
   resp = PH)

# one categorical variable
p1 <- plot_bars(data_g, GEN, PH)

p2 <- plot_bars(data_g, GEN, PH,
  n.dodge = 2, # two rows for x labels
  y.expand = 0.1, # expand y scale
  errorbar = FALSE, # remove errorbar
  color.bar = "red", # color of bars
  fill.bar = alpha_color("cyan", 75), # create a transparent color
  lab.bar = letters[1:13], # add labels
  plot_theme = ggplot2::theme_gray()) #change plot themes

arrange_ggplot(p1, p2)
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

```r
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 <tiago.olivoto@gmail.com>

Examples

```r
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,
  FG,
  SG,
  by = NULL,
  use = "cor",
  test = "Bartlett",
  prob = 0.05,
  center = TRUE,
  stdscores = FALSE,
  verbose = TRUE,
  collinearity = TRUE
)

Arguments
.data The data to be analyzed. It can be a data frame (possible with grouped data passed from group_by()).
FG, SG 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 compute the function by. It is a shortcut to group_by(). To compute the statistics by more than one grouping variable use that function.
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 a grouped data passed from group_by() then the results will be returned into a list-column of data frames.

- **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

Examples

```r
c1 <- can_corr(data_ge2, 
   FG = c(PH, EH, EP), 
   SG = c(EL, ED, CL, CD, CW, KW, NR))

# Canonical correlations for each environment
c3 <- data_ge2 %>%
   can_corr(FG = c(PH, EH, EP), 
            SG = c(EL, ED, CL, CD, CW, KW, NR),
```
clustering(by = ENV,
    verbose = FALSE)

clustering

Clustering analysis

Description

Performs clustering analysis with selection of variables.

Usage

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

Arguments

.data The data to be analyzed. It can be a data frame, possible with grouped data passed from .
    group_by().
... 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 compute the function by. It is a shortcut to group_by().
    To compute the statistics by more than one grouping variable use that function.
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 cophenetic 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, cofgrap 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
Examples

library(metan)

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

# Based on the mean for each genotype
mean_gen <-
data_ge2 %>%
  means_by(GEN) %>%
  column_to_rownames("GEN")

d2 <- clustering(mean_gen)

# Select variables for compute the distances
d3 <- clustering(mean_gen, selvar = TRUE)

# Compute the distances with standardized data
# Define 4 clusters
d4 <- clustering(data_ge,
  by = ENV,
  scale = TRUE,
  nclust = 4)

---

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)

Arguments

.data The data to be analyzed. It must be a symmetric correlation matrix, or a data frame, possible with grouped data passed from group_by().

... 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 compute the function by. It is a shortcut to group_by(). To compute the statistics by more than one grouping variable use that function.
If a correlation matrix is provided, then $n$ is the number of objects used to compute the correlation coefficients.

**Value**

If `.data` is a grouped data passed from `group_by()` then the results will be returned into a list-column of data frames.

- **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.
- **ncorhigh** Number of correlation greater than |0.8|.
- **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**


**Examples**

```r
# Using the correlation matrix
library(metan)
cor_iris <- cor(iris[,1:4])
n <- nrow(iris)
col_diag <- colindiag(cor_iris, n = n)
```
```r
# Using a data frame
col_diag_gen <- data_ge2 %>%
  group_by(GEN) %>%
colindiag()

# Diagnostic by levels of a factor
# For variables with "N" in variable name
col_diag_gen <- data_ge2 %>%
  group_by(GEN) %>%
colindiag(contains("N"))
```

---

### 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.
- **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

```r
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

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

Arguments

- `.data` The data to be analyzed. It can be a data frame (possible with grouped data passed from `group_by()`) 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 compute the function by. It is a shortcut to `group_by()`. To compute the statistics by more than one grouping variable use that function.
- `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:

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

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


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, 
               verbose = FALSE)
CI2

---

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
corr_plot

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 = 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
c coworkers, 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
**corr_plot**

- `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)

# Choosing 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:

\[ n = \left[\frac{CI_w}{0.45304r \times 2.25152}\right]^{-0.50089} \]

where \( CI_w \) is desired confidence interval and \( r \) is the correlation coefficient.

Author(s)

Tiago Olivoto <tiagoolivoto@gmail.com>

References


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), "P1_a", "P1_f", "P1_u" (Superiority indexes for all, favorable and unfavorable environments, respectively), "Ga1" (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>
covcor_design

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 data to be analyzed. It can be a data frame, possible with grouped data passed from group_by().
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 compute the function by. It is a shortcut to group_by(). To compute the statistics by more than one grouping variable use that function.
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 a grouped data passed from `group_by()` then the results will be returned into a list-column of data frames.

**Author(s)**

Tiago Olivoto <tiagoolivoto@gmail.com>

**Examples**

```r
library(metan)
# List of matrices
data <- subset(data_ge2, ENV == 'A1')
mats <- 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, 
                      rep = REP, 
                      resp = c(PH, EH, CD, CL), 
                      by = ENV,  
                      type = "rcov")
```

**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. AT LEAST THREE REPLICATES ARE REQUIRED TO PERFORM THE CROSS-VALIDATION.

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, except the error, 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 = 'RCBD' 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.
IMPORTANT: If the data set is unbalanced (i.e., any genotype missing in any environment) the function will return an error. An error is also observed if any combination of genotype-environment has a different number of replications than observed in the trial.

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


See Also

cv_ammi,cv_blup

Examples

```r
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

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

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>.data</td>
<td>The dataset containing the columns related to Environments, Genotypes, replication/block and response variable(s).</td>
</tr>
<tr>
<td>env</td>
<td>The name of the column that contains the levels of the environments.</td>
</tr>
<tr>
<td>gen</td>
<td>The name of the column that contains the levels of the genotypes.</td>
</tr>
<tr>
<td>rep</td>
<td>The name of the column that contains the levels of the replications/blocks. AT LEAST THREE REPLICATES ARE REQUIRED TO PERFORM THE CROSS-VALIDATION.</td>
</tr>
<tr>
<td>resp</td>
<td>The response variable.</td>
</tr>
<tr>
<td>nboot</td>
<td>The number of resamples to be used in the cross-validation. Defaults to 200.</td>
</tr>
<tr>
<td>block</td>
<td>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, except the error, are assumed to be fixed.</td>
</tr>
<tr>
<td>design</td>
<td>The experimental design used in each environment. Defaults to RCBD (Randomized complete Block Design). For Completely Randomized Designs inform design = 'CRD'.</td>
</tr>
<tr>
<td>verbose</td>
<td>A logical argument to define if a progress bar is shown. Default is TRUE.</td>
</tr>
</tbody>
</table>
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.

IMPORTANT: If the data set is unbalanced (i.e., any genotype missing in any environment) the function will return an error. An error is also observed if any combination of genotype-environment has a different number of replications than observed in the trial.

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


See Also

cv_ammi,cv_blup

Examples

library(metan)
model <- cv_ammif(data_ge,
cv_blup

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**

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

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>.data</code></td>
<td>The dataset containing the columns related to Environments, Genotypes, replication/block and response variable(s).</td>
</tr>
<tr>
<td><code>env</code></td>
<td>The name of the column that contains the levels of the environments.</td>
</tr>
<tr>
<td><code>gen</code></td>
<td>The name of the column that contains the levels of the genotypes.</td>
</tr>
<tr>
<td><code>rep</code></td>
<td>The name of the column that contains the levels of the replications/blocks. <strong>AT LEAST THREE REPLICATES ARE REQUIRED TO PERFORM THE CROSS-VALIDATION.</strong></td>
</tr>
<tr>
<td><code>resp</code></td>
<td>The response variable.</td>
</tr>
<tr>
<td><code>block</code></td>
<td>Defaults to <code>NULL</code>. 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 <strong>Details</strong>.</td>
</tr>
</tbody>
</table>
The number of resamples to be used in the cross-validation. Defaults to 200

The effects of the model assumed to be random. See Details for more information.

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 in each environment assuming genotype and genotype-environment interaction as random effects. Environments and blocks nested within environments are assumed to fixed factors.

- **Model 2**: block = NULL and random = "env". This model considers a Randomized Complete Block Design in each environment treating environment, genotype-environment interaction, and blocks nested 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 in each environment assuming a random-effect model, i.e., all effects (genotypes, environments, genotype-vs-environment interaction and blocks nested within environments) are assumed to be random factors.

- **Model 4**: block is not NULL and random = "gen". This model considers an alpha-lattice design in each environment assuming genotype, genotype-environment interaction, and incomplete blocks nested within complete replicates as random to make use of inter-block information (Mohring et al., 2015). Complete replicates nested within environments and environments are assumed to be fixed factors.

- **Model 5**: block is not NULL and random = "env". This model considers an alpha-lattice design in each environment assuming genotype as fixed. All other sources of variation (environment, genotype-environment interaction, complete replicates nested within environments, and incomplete blocks nested within replicates) are assumed to be random factors.

- **Model 6**: block is not NULL and random = "all". This model considers an alpha-lattice design in each environment assuming all effects, except the intercept, as random factors.

**IMPORTANT**: An error is returned if any combination of genotype-environment has a different number of replications than observed in the trial.

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.
**data_alpha**

**Author(s)**

Tiago Olivoto <tiagoolivoto@gmail.com>

**References**


**See Also**

cv_ammi, cv_ammif

**Examples**

```r
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**

---

**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
Description

This dataset contains data on 15 traits assessed in 13 maize hybrids growing in 4 environments. The experimental design was a RCBD with 3 blocks and 1 replication 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

- desc_stat() Computes the most used measures of central tendency, position, and dispersion.
- desc_wider() is useful to put the variables in columns and grouping variables in rows. The table is filled with a statistic chosen with the argument stat.

Usage

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

```r
desc_wider(.data, which)
```

Arguments

- `.data` The data to be analyzed. It can be a data frame (possible with grouped data passed from `group_by()`) or a numeric vector. For `desc_wider()`, `.data` is an object of class `desc_stat`.
- `...` A single variable name or a comma-separated list of unquoted variables names. If no variable is informed, all the numeric variables from `.data` will be used. Select helpers are allowed.
- `by` One variable (factor) to compute the function by. It is a shortcut to `group_by()`.
- `stats` The descriptive statistics to show. This is used to filter the output after computation. Defaults to "main" (cv, max, mean, median, min, sd.amo, se, ci). Other allowed values are "all" to show all the statistics, "robust" to show robust statistics, "quantile" to show quantile statistics, or chose one (or more) of the following:
  - "av.dev": average deviation.
  - "ci": 95 percent confidence interval of the mean.
  - "cv": coefficient of variation.
  - "iqr": interquartile range.
• "gmean": geometric mean.
• "hmean": 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", "q25", "q75", "q97.5": the percentile 2.5 \ quartile, third quartile, and percentile 97.5 \ range: The range of data.
• "sd. amo", "sd. pop": the sample and population standard deviation.
• "se": the standard error of the mean.
• "skew": 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", "var. pop": the sample and population variance.

Use a names to select the statistics. For example, `stats = c("median, mean, cv, n")`. Note that the statistic names are not case-sensitive. Both comma or space can be used as separator.

`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? Defaults to `FALSE`.

`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`.

`which` A statistic to fill the table.

**Value**

- `desc_stats()` returns a tibble with the statistics in the columns and variables (with possible grouping factors) in rows.
- `desc_wider()` returns a tibble with variables in columns and grouping factors in rows.

**Author(s)**

Tiago Olivoto <tiagoolivoto@gmail.com>
Examples

library(metan)

# Example 1: main statistics (coefficient of variation, maximum, #
# mean, median, minimum, sample standard deviation, standard  #
# error and confidence interval of the mean) for all numeric     #
# variables in data                                         #
#===============================================================#

desc_stat(data_ge2)

#===============================================================#

# Example 2: robust statistics using a numeric vector as input #
# data                                                       #
#===============================================================#

vect <- data_ge2$TKW

desc_stat(vect, stats = "robust")

#===============================================================#

# Example 3: Select specific statistics. In this example, NAs #
# are removed before analysis with a warning message        #
#===============================================================#

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

#===============================================================#

# Example 4: Select specific variables and compute statistics by#
# levels of a factor variable (GEN)                        #
#===============================================================#

stats <-

desc_stat(data_ge2,
          EP, EL, EH, ED, PH, CD,
          by = GEN)

stats

# To get a 'wide' format with the maximum values for all variables
desc_wider(stats, max)

#===============================================================#

# Example 5: Compute all statistics for all numeric variables #
# by two or more factors. Note that group_by() was used to pass #
# grouped data to the function desc_stat()                  #
#===============================================================#

data_ge2 %>%
  group_by(ENV, GEN) %>%
desc_stat()
doo

Alternative to dplyr::do for doing anything

Description

Provides an alternative to the dplyr::do() using nest(), mutate() and map() to apply a function to a grouped data frame.

Usage

doo(.data, .fun, ...)

Arguments

- .data: a (grouped) data frame
- .fun: A function, formula, or atomic vector.
- ...: Additional arguments passed on to .fun

Details

If the applied function returns a data frame, then the output will be automatically unnested. Otherwise, the output includes the grouping variables and a column named "data" which is a "list-columns" containing the results for group combinations.

Value

a data frame

Author(s)

Tiago Olivoto <tiagoolivoto@gmail.com>

Examples

library(metan)
# Head the first two lines of each environment
data_ge2 %>%
  group_by(ENV) %>%
  doo(~head(., 2))

# Genotype analysis for each environment using 'gafem()'
# variable PH
data_ge2 %>%
  group_by(ENV) %>%
  doo(~gafem(., GEN, REP, PH, verbose = FALSE))
ecovalence

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


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**

```r
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**

Robertson (1959) proposed the partition of the mean square of the genotype-environment interaction (MS_GE) into single (S) and complex (C) parts, where

\[ S = \frac{1}{2} \left( \sqrt{Q_1} - \sqrt{Q_2} \right)^2 \]

and

\[ C = (1 - r) \sqrt{Q_1 - Q_2} \]

being \( r \) the correlation between the genotype's average in the two environments; and \( Q_1 \) and \( Q_2 \) 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

\[ C = \sqrt{(1 - r)^2 \times Q_1 \times Q_2} \]

**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


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.
find_outliers

**Value**

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

- **data** The data (BLUPS) used to compute the index.
- **eigen** The eigenvalues and explained variance for each axis.
- **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.differential** A list with the selection differential for each ideotype design.

**Author(s)**

Tiago Olivoto <tiagoolivoto@gmail.com>

**References**


**Examples**

```r
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'))
```

---

**find_outliers**

*Find possible outliers in a dataset*

**Description**

Find possible outliers in the dataset.
Usage

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

Arguments

.data The data to be analyzed. Must be a data frame or an object of class split_factors.
var The variable to be analyzed.
by One variable (factor) to compute the function by. It is a shortcut to \texttt{\textbf{group\_by()}}. To compute the statistics by more than one grouping variable use that function.
plots If \texttt{\textbf{TRUE}}, then histograms and boxplots are shown.
coef The multiplication coefficient, defaults to 1.5. For more details see \texttt{?boxplot.stat}.
verbose If \texttt{\textbf{verbose = TRUE}} then some results are shown in the console.
plot_theme The graphical theme of the plot. Default is \texttt{\textbf{plot\_theme = theme\_metan()}}. For more details, see \texttt{theme}.  

Author(s)

Tiago Olivoto \texttt{<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)

\begin{tabular}{ll}
\hline
Fox & \textit{Fox's stability function} \\
\hline
\end{tabular}

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, resp, verbose = TRUE)

Arguments

.data The dataset containing the columns related to Environments, Genotypes, replications/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 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


Examples

library(metan)
out <- Fox(data_ge2, ENV, GEN, PH)
print(out)
gafem

Genotype analysis by fixed-effect models

Description

One-way analysis of variance of genotypes conducted in both randomized complete block and alpha-lattice designs.

Usage

gafem(.data, gen, rep, resp, prob = 0.05, block = NULL, 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.
- `prob` The error probability. Defaults to 0.05.
- `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, except the error, are assumed to be fixed.** Use the function `gamem` to analyze a one-way trial with mixed-effect models.
- `verbose` Logical argument. If `verbose = FALSE` the code are run silently.

Details

gafem 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 \( i \)th genotype in the \( j \)th block; \( m \) is the grand mean (fixed); \( g_i \) is the effect of the \( i \)th genotype; \( r_j \) is the effect of the \( j \)th replicate; 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 \( y_{ijk} \). All effects, except the random error are assumed to be fixed.
Value

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

• **anova**: The one-way ANOVA table.
• **model**: The model with `lm`.
• **augment**: Information about each observation in the dataset. This includes predicted values in the `fitted` column, residuals in the `resid` column, standardized residuals in the `stdres` column, the diagonal of the ‘hat’ matrix in the `hat` column, and standard errors for the fitted values in the `se.fit` column.
• **hsd**: The Tukey’s ‘Honest Significant Difference’ for genotype effect.
• **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 loser winner genotype, MaxGEN, the winner genotype.

Author(s)

Tiago Olivoto <tiagoolivoto@gmail.com>

References


See Also

get_model_data gamem

Examples

library(metan)
# RCBD
rcbd <- gafem(data_g,
gen = GEN,
rep = REP,
resp = c(PH, ED, EL, CL, CW))
# Fitted values
get_model_data(rcbd)

# ALPHA-LATTICE DESIGN
alpha <- gafem(data_alpha,
gen = GEN,
rep = REP,
block = BLOCK,
resp = YIELD)
# Fitted values
get_model_data(alpha)
**gai**

*Geometric adaptability index*

**Description**

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

\[
GAI = \sqrt[n]{Y_1 \cdot Y_2 \cdot \cdots \cdot Y_i}
\]

**Usage**

```r
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**

Examples

```r
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**

```r
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 $i$th 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
- **ranef**: The random effects of the model
- **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**


**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")

# random effects
get_model_data(rcbd, "ranef")

# Predicted values
predict(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)

# Genetic parameters
get_model_data(alpha, "genpar")

# Random effects
get_model_data(alpha, "ranef")

---

gamem_met

**Genotype-environment analysis by mixed-effect models**
Description
Genotype analysis in multi-environment trials using mixed-effect or random-effect models.

Usage

```
gamem_met(
  .data,  
  env, 
  gen, 
  rep, 
  resp, 
  block = NULL, 
  random = "gen", 
  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 a vector of variables may be used. For example `resp = c(var1, var2, var3)`.
- `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).
- `random` The effects of the model assumed to be random. Defaults to `random = "gen"`. See Details to see the random effects assumed depending on the experimental design of the trials.
- `prob` The probability for estimating confidence interval for BLUP’s prediction.
- `verbose` Logical argument. If `verbose = FALSE` the code will run silently.

Details
The nature of the effects in the model is chosen with the argument `random`. By default, the experimental design considered in each environment is a randomized complete block design. If `block` is informed, a resolvable alpha-lattice design (Patterson and Williams, 1976) is implemented. The following six models can be fitted depending on the values of `random` and `block` arguments.

- **Model 1**: `block = NULL` and `random = "gen"` (The default option). This model considers a Randomized Complete Block Design in each environment assuming genotype and genotype-environment interaction as random effects. Environments and blocks nested within environments are assumed to fixed factors.
• **Model 2**: block = NULL and random = "env". This model considers a Randomized Complete Block Design in each environment treating environment, genotype-environment interaction, and blocks nested 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 in each environment assuming a random-effect model, i.e., all effects (genotypes, environments, genotype-vs-environment interaction and blocks nested within environments) are assumed to be random factors.

• **Model 4**: block is not NULL and random = "gen". This model considers an alpha-lattice design in each environment assuming genotype, genotype-environment interaction, and incomplete blocks nested within complete replicates as random to make use of inter-block information (Mohring et al., 2015). Complete replicates nested within environments and environments are assumed to be fixed factors.

• **Model 5**: block is not NULL and random = "env". This model considers an alpha-lattice design in each environment assuming genotype as fixed. All other sources of variation (environment, genotype-environment interaction, complete replicates nested within environments, and incomplete blocks nested within replicates) are assumed to be random factors.

• **Model 6**: block is not NULL and random = "all". This model considers an alpha-lattice design in each environment assuming all effects, except the intercept, as random factors.

**Value**

An object of class waasb with the following items for each variable:

- **fixed** Test for fixed effects.
- **random** Variance components for random effects.
- **LRT** The Likelihood Ratio Test for the random effects.
- **BLUPgen** The random effects and estimated BLUPS for genotypes (If random = "gen" or random = "all")
- **BLUPenv** The random effects and estimated BLUPS for environments, (If random = "env" or random = "all").
- **BLUPint** The random effects and estimated BLUPS of all genotypes in all environments.
- **MeansGxE** The phenotypic means of genotypes in the environments.

**Details** A list summarizing the results. The following information are shown: Nenv, the number of environments in the analysis; Ngen the number of genotypes in the analysis; Mean the grand mean; SE the standard error of the mean; SD the standard deviation. CV the coefficient of variation of the phenotypic means, estimating WAASB, Min the minimum value observed (returning the genotype and environment), Max the maximum value observed (returning the genotype and environment); 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 tibble with the genetic parameters (if random = "gen" or random = "all") with the following columns: Phenotypic variance the phenotypic variance; Heritability the broad-sense heritability; GER2 the coefficient of determination of the interaction effects; Heribatility of means the heritability on the mean basis; Accuracy the selective accuracy;
rge the genotype-environment correlation; CVg the genotypic coefficient of variation; CVr the residual coefficient of variation; CV ratio the ratio between genotypic and residual coefficient of variation.

- **residuals** The residuals of the model.

**Author(s)**

Tiago Olivoto <tiagoolivoto@gmail.com>

**References**


**See Also**

mtsia waas get_model_data plot_scores

**Examples**

```r
library(metan)
#===============================================================#
# Example 1: Analyzing all numeric variables assuming genotypes #
# as random effects #
#===============================================================#
model <- gamem_met(data_ge,
    env = ENV,
    gen = GEN,
    rep = REP,
    resp = everything())
# Distribution of random effects (first variable)
plot(model, type = "re")

# Genetic parameters
get_model_data(model, "genpar")

#===============================================================#
# Example 2: Unbalanced trials #
# assuming all factors as random effects #
#===============================================================#
un_data <- data_ge %>%
    remove_rows(1:3) %>%
    droplevels()
```
model2 <- gamem_met(un_data,
  env = ENV,
  gen = GEN,
  rep = REP,
  random = "all",
  resp = GY)
get_model_data(model2)

get_model_data
Get data from a model easily

Description
- get_model_data() 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.
- gmd() Is a shortcut to get_model_data.

Usage
get_model_data(x, what = NULL, type = "GEN", verbose = TRUE)
gmd(x, what = NULL, type = "GEN", verbose = TRUE)

Arguments
x An object created with the functions AMMI_indexes, anova_ind, anova_joint, ecovervalence, Fox, gai, gamem, gafem, 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.
verbose Logical argument. If verbose = FALSE the code will run silently.

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** `anova_ind`:

• "MEAN" The mean value of the variable
• "MSG", "FCG", "PFG" The mean square, F-calculated and P-values for genotype effect, respectively.
• "MSB", "FCB", "PFB" The mean square, F-calculated and P-values for block effect in randomized complete block design.
• "MSCR", "FCR", "PFCR" The mean square, F-calculated and P-values for complete replicates in alpha lattice design.
• "MSIB_R", "FCIB_R", "PFIB_R" The mean square, F-calculated and P-values for incomplete blocks within complete replicates, respectively (for alpha lattice design only).
• "MSE" The mean square of error.
• "CV" The coefficient of variation.
• "h2" The broad-sense heritability
• "MSE" The accuracy of selection (square root of h2).

**Objects of class** `anova_joint` or `gafem`:

• "Sum Sq" Sum of squares.
• "Mean Sq" Mean Squares.
• "F value" F-values.
• "Pr(>F)" P-values.
• ".fitted" Fitted values (default).
• ".resid" Residuals.
• ".stdresid" Standardized residuals.
• ".se.fit" Standard errors of the fitted values.
• "details" Details.

**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 slope 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 stability variance (default).
• "rShukaVar" Rank for Shukla's stability 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_ami`:

• "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 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 axis.
• "WAASBY" The superiority index WAASBY.
• "OrWAASBY" The ranking regarding the superiority index.

**Objects of class** `waasb`:

• "blupg" For genotype’s predicted mean.
• "blupge" for genotype-vs-environment’s predicted mean.
• "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.
• "ranef" Random effects.

**Objects of class** `gamem`:

• "blupg" For genotype’s predicted mean.
• "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.
• "ranef" 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**


get_model_data

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


See Also

AMMI_indexes, anova_ind, anova_joint, ecovalence, Fox, gai, gamem, gafem, ge_means, ge_reg, performs_ammi, Resende_indexes, Shukla, superiority, waas, waasb

Examples

library(metan)

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 = "WAAS")

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

#################### 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 before 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): 

\[
Z_{ij} = \frac{Y_{ij} - \bar{Y}_j}{S_j}
\]

cluster What should be clustered? Defaults to cluster = "env" (cluster environments). To cluster the genotypes use cluster = "gen".

nclust The number of clusters 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**


**Examples**

```r
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**

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

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.
- SE: The standard error of the mean.
- SD: The standard deviation.
- CV: The coefficient of variation.
- Min, Max: The minimum and maximum value, indicating the genotype and environment of occurrence.
- 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, 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.

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

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

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, \( \text{resp} = c(\text{var1}, \text{var2}, \text{var3}). \)

.mineval
The minimum value so that an eigenvector is retained in the factor analysis.

.verbose
Logical argument. If \( \text{verbose} = \text{FALSE} \) 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.

cormat.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


See Also

superiority, ecovalence, ge_stats, ge_reg
Examples

```r
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**

```r
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

```r
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")
```

---

**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**

```r
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

```r
library(metan)
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

```r
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 following 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


See Also

superiority, ecovalence, ge_stats

Examples

library(metan)
reg <- ge_reg(data_ge2,
     env = ENV,
     gen = GEN,
     rep = REP,
     resp = PH)
plot(reg)
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" (Annichiariico's genotypic confidence index for all, favorable and unfavorable environments, respectively, calling Annichiario internally), "Ecoval" (Wricke's ecosalence, 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


Examples

```r
library(metan)

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

<table>
<thead>
<tr>
<th>ge_winners</th>
<th>Genotype-environment winners</th>
</tr>
</thead>
</table>

Description

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

Usage

```r
gw <- 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 \texttt{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 \texttt{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 are ranked from minimum to maximum. Use a comma-separated vector of names. For example, \texttt{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 \texttt{type = "ranks"}).

Author(s)

Tiago Olivoto <tiagoolivoto@gmail.com>

Examples

```r
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**

```r
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).
- `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 the 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.

Arguments passed to the function `impute_missing_val()` for imputation of missing values in case of unbalanced data.

**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**


**Examples**

```r
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, var = "ED")

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

table <- make_mat(data_ge, GEN, ENV, GY) %>% round(2)
table
make_long(table) %>%
gge(ENV, GEN, Y) %>%
plot()

---

**gtb**

*Genotype by trait biplot*

### Description

Produces a genotype-by-trait biplot model. From a genotype by environment by trait three-way table, genotype-by-trait tables in any single environment, across all environments, or across a subset of the environments can be generated and visually studied using biplots. The model for biplot analysis of genotype by trait data is the singular value decomposition of trait-standardized two-way table.

### Usage

```r
gtb(.data, gen, resp, centering = "trait", scaling = "sd", svp = "trait")
```

### Arguments

- **.data**: The dataset containing the columns related to Genotypes and the response variable(s).
- **gen**: The name of the column that contains the levels of the genotypes.
- **resp**: The response variables, i.e., `resp = c(var1, var2, var3)`. Select helpers can also be used.
- **centering**: The centering method. Must be one of the 'none | 0', for no centering; 'global | 1', for global centered (T+G+GT); 'trait | 2' (default), for trait-centered (G+GT); or 'double | 3', for double centred (GT). A biplot cannot be produced with models produced without centering.
- **scaling**: The scaling method. Must be one of the 'none | 0', for no scaling; or 'sd | 1' (default), where each value is divided by the standard deviation of its corresponding trait (column). This will put all traits roughly the same range 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); 'trait | 2', default, (The singular value is entirely partitioned into the trait eigenvectors, also called column metric preserving); or 'symmetrical | 3' (The singular value is symmetrically partitioned).
into the genotype and the trait 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 traits).

**Value**

The function returns a list of class `gge` that is compatible with the function `plot()` used in `gge()`.

- `coordgen` The coordinates for genotypes for all components.
- `coordenv` The coordinates for traits for all components.
- `eigenvalues` The vector of eigenvalues.
- `totalvar` The overall variance.
- `labelgen` The name of the genotypes.
- `labelenv` The names of the traits.
- `labelaxes` The axes labels.
- `gt_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 traits are comparable when singular value partitioning is set to 'genotype' or 'trait'.
- `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 traits.
- `scale_var` The scaling vector when the scaling method is 'sd'.

**Author(s)**

Tiago Olivoto <tiagoolivoto@gmail.com>

**References**


**Examples**

```r
library(metan)
# GT biplot for all numeric variables
mod <- gtb(data_ge2, GEN, resp = contains("E"))
plot(mod)
```
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, 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.
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

Examples

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

impute_missing_val

Description
Impute the missing entries of a matrix with missing values using different algorithms. See Details section for more details

Usage

impute_missing_val(
  .data,
  naxis = 1,
  algorithm = "EM-SVD",
  tol = 1e-10,
  max_iter = 1000,
  simplified = FALSE,
  verbose = TRUE
)

Arguments

.data A matrix to impute the missing entries. Frequently a two-way table of genotype means in each environment.
naxis The rank of the Singular Value Approximation. Defaults to 1.
algorithm The algorithm to impute missing values. Defaults to "EM-SVD". Other possible values are "EM-AMMI" and "colmeans". See Details section.
tol The convergence tolerance for the algorithm.
max_iter The maximum number of steps to take. If max_iter is achieved without convergence, the algorithm will stop with a warning.
simplified Valid argument when algorithm = "EM-AMMI". IF FALSE (default), the current effects of rows and columns change from iteration to iteration. If TRUE, the general mean and effects of rows and columns are computed in the first iteration only, and in next iterations uses these values.
verbose Logical argument. If verbose = FALSE the code will run silently.
Details

EM-AMMI algorithm

The EM-AMMI algorithm completes a data set with missing values according to both main and interaction effects. The algorithm works as follows (Gauch and Zobel, 1990):

1. The initial values are calculated as the grand mean increased by main effects of rows and main effects of columns. That way, the matrix of observations is pre-filled in.
2. The parameters of the AMMI model are estimated.
3. The adjusted means are calculated based on the AMMI model with naxis principal components.
4. The missing cells are filled with the adjusted means.
5. The root mean square error of the predicted values (RMSE_p) is calculated with the two last iteration steps. If RMSE_p > tol, the steps 2 through 5 are repeated. Declare convergence if RMSE_p < tol. If max_iter is achieved without convergence, the algorithm will stop with a warning.

EM-SVD algorithm

The EM-SVD algorithm impute the missing entries using a low-rank Singular Value Decomposition approximation estimated by the Expectation-Maximization algorithm. The algorithm works as follows (Troyanskaya et al., 2001).

1. Initialize all NA values to the column means.
2. Compute the first naxis terms of the SVD of the completed matrix
3. Replace the previously missing values with their approximations from the SVD
4. The root mean square error of the predicted values (RMSE_p) is calculated with the two last iteration steps. If RMSE_p > tol, the steps 2 through 3 are repeated. Declare convergence if RMSE_p < tol. If max_iter is achieved without convergence, the algorithm will stop with a warning.

colmeans algorithm

The colmeans algorithm simply impute the missing entries using the column mean of the respective entire. Thus, there is no iterative process.

Value

An object of class imv with the following values:

- .data The imputed matrix
- pc_ss The sum of squares representing variation explained by the principal components
- iter The final number of iterations.
- Final_RMSE The maximum change of the estimated values for missing cells in the last step of iteration.
- final_axis The final number of principal component axis.
- convergence Logical value indicating whether the modern converged.
References


Examples

```r
library(metan)
mat <- (1:20) %*% t(1:10)
mat
# 10% of missing values at random
miss_mat <- random_na(mat, prop = 10)
miss_mat
mod <- impute_missing_val(miss_mat)
mod$.data
```

---

**inspect**

*Check for common errors 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**

```r
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

```r
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)
```

---

**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_balanced_trial  
Check if a data set is balanced

Description

Check if a data set coming from multi-environment trials is balanced, i.e., all genotypes are in all environments.

Usage

is_balanced_trial(.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.

 Value

A logical value

 Author(s)

Tiago Olivoto <tiagoolivoto@gmail.com>

 Examples

unb <- data_ge %>%
   remove_rows(1:3) %>%
   droplevels()
is_balanced_trial(data_ge, ENV, GEN, GY)
is_balanced_trial(unb, ENV, GEN, GY)

Description

• plot_lines() Creates a line plot based on one quantitative factor and one numeric variable. It can be used to show the results of a one-way trial with quantitative treatments.

• plot_factlines() Creates a line plot based on: one categorical and one quantitative factor and one numeric variable. It can be used to show the results of a two-way trial with qualitative-quantitative treatment structure.

Usage

plot_lines(
  .data,
  x,
  y,
  fit,
  level = 0.95,
  confidence = TRUE,
xlab = NULL,
ylab = NULL,
n.dodge = 1,
check.overlap = FALSE,
col = "red",
alpha = 0.2,
size.shape = 1.5,
size.line = 1,
size.text = 12,
fontfam = "sans",
plot_theme = theme_metan()
)

plot_factlines(
  .data,
  x,
  y,
  group,
  fit,
  level = 0.95,
  confidence = TRUE,
  xlab = NULL,
  ylab = NULL,
  n.dodge = 1,
  check.overlap = FALSE,
  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, y            The variables to be mapped to the x and y axes, respectively.
fit             The polynomial degree to use. It must be between 1 (linear fit) to 4 (fourth-order polynomial regression). In plot_factlines(), if fit is a length 1 vector, i.e., 1, the fitted curves of all levels in group will be fitted with polynomial degree fit. To use a different polynomial degree for each level in group, use a numeric vector with the same length of the variable in group.
level           The confidence level. Defaults to 0.95.
confidence      Display confidence interval around smooth? (TRUE by default)
xlab, ylab  The labels of the axes x and y, respectively. Defaults to NULL.
n.dodge  The number of rows that should be used to render the x labels. This is useful for displaying labels that would otherwise overlap.
check.overlap  Silently remove overlapping labels, (recursively) prioritizing the first, last, and middle labels.
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.
group  The grouping variable. Valid for plot_factlines() only.
legend.position  Valid argument for plot_factlines. The position of the legend. Defaults to 'bottom'.
grid  Valid argument for plot_factlines. Logical argument. If TRUE then a grid will be created.
scales  Valid argument for plot_factlines. If grid = TRUE scales controls how the scales are in the plot. Possible values are 'free' (default), 'fixed', 'free_x' or 'free_y'.

Value
An object of class gg, ggplot.

Author(s)
Tiago Olivoto <tiagoolivoto@gmail.com>

See Also
plot_bars and plot_factbars

Examples

library(metan)
# One-way line plot
df1 <- data.frame(group = "A",
               x = c(0, 100, 200, 300, 400),
               y = c(3.2, 3.3, 4.0, 3.8, 3.4))
plot_lines(df1, x, y, fit = 2)

# Two-way line plot
df2 <- data.frame(group = "B",
  x = c(0, 100, 200, 300, 400),
  y = c(3.2, 3.3, 3.7, 3.9, 4.1))
facts <- rbind(df1, df2)

p1 <- plot_factlines(facts, x, y, group = group, fit = 1)
p2 <- plot_factlines(facts,
  x = x,
  y = y,
  group = group,
  fit = c(2, 1),
  confidence = FALSE)
arrange_ggplot(p1, p2)

---

### 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**

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

**Arguments**

- **.data**
  
  The data to be analyzed. It must be a symmetric correlation matrix or a data frame, possible with grouped data passed from `group_by()`.

- **...**
  
  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 compute the function by. It is a shortcut to `group_by()`.
  To compute the statistics by more than one grouping variable use that function.

- **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"`.

**Value**

If `.data` is a grouped data passed from `group_by()` then the results will be returned into a list-column of data frames, containing:

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

```r
library(metan)
partial1 <- lpcor(iris)

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

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

# Select all numeric variables and compute the partial correlation
# For each level of Species
partial4 <- lpcor(iris, by = Species)
```

---

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

```r
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

```r
library(metan)
library(dplyr)

# Compute the mean for genotypes
means <- means_by(data_ge, GEN) %>%
  column_to_rownames("GEN")

# Compute the covariance matrix
covmat <- cov(means)

# Compute the distance
dist <- mahala(means, covmat)

# Dendrogram
dend <- dist %>%
  as.dist() %>%
  hclust() %>%
  as.dendrogram()
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**

```r
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, possible with grouped data passed from `group_by()`.
- `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.
The response variables. For example `resp = c(var1, var2, var3)`.

The experimental design. Must be RCBD or CRD.

One variable (factor) to compute the function by. It is a shortcut to `group_by()`.
To compute the statistics by more than one grouping variable use that function.

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`.

A symmetric matrix with the Mahalanobis’ distance. If `.data` is a grouped data passed from `group_by()` then the results will be returned into a list-column of data frames.

Tiago Olivoto <tiagoolivoto@gmail.com>

Examples

```r
library(metan)
maha <- mahala_design(data_g, 
  gen = GEN, 
  rep = REP, 
  resp = everything(), 
  return = "covmat")
# Compute one distance for each environment (all numeric variables)
maha_group <- mahala_design(data_ge, 
  gen = GEN, 
  rep = REP, 
  resp = 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')
```

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

```r
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

```r
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")
```

---

### Description

This function helps 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 contain 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

```r
library(metan)
matrix <- data_ge %>% make_mat(row = GEN, col = ENV, val = GY)
matrix
# standart error of mean

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

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>
mgidi

Genotype-Ideotype Distance Index

Description

Computes the multi-trait genotype-ideotype distance index (MGIDI). MGIDI can be seen as the multi-trait stability index (Olivoto et al., 2019) computed with weight for mean performance equals to 100.

Usage

mgidi(.data, SI = 15, mineval = 1, ideotype = NULL, verbose = TRUE)

Arguments

.data An object fitted with gamem, gamem_met, or a two-way table with BLUPs for genotypes in each trait (genotypes in rows and traits in columns). In the last case, row names must contain the genotypes names.

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.

ideotype A vector of length nvar where nvar is the number of variables used to plan the ideotype. Use ‘h’ to indicate the variables with increase in selection or ‘l’ to indicate the variables with reduction in selection. For example, ideotype = c("h,h,l,h,l").

verbose If verbose = TRUE (Default) then some results are shown in the console.

Value

An object of class mgidi 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.
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>
non_collinear_vars

Select a set of predictors with minimal multicollinearity

Description

Select a set of predictors with minimal multicollinearity using the variance inflation factor (VIF) as criteria to remove collinear variables. The algorithm will: (i) compute the VIF value of the correlation matrix containing the variables selected in . . .; (ii) arrange the VIF values and delete the variable with the highest VIF; and (iii) iterate step ii until VIF value is less than or equal to max_vif.

Usage

```r
non_collinear_vars(
  .data,
  ..., 
  max_vif = 10,
  missingval = "pairwise.complete.obs"
)
```

Examples

```r
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()
```

References

pairs_mantel

Arguments

.data The data set containing the variables.

... Variables to be submitted to selection. 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.

max_vif The maximum value for the Variance Inflation Factor (threshold) that will be accepted in the set of selected predictors.

missingval How to deal with missing values. For more information, please see cor().

Value

A data frame showing the number of selected predictors, maximum VIF value, condition number, determinant value, selected predictors and removed predictors from the original set of variables.

Examples

library(metan)
# All numeric variables
non_collinear_vars(data_ge2)

# Select variables and choose a VIF threshold to 5
non_collinear_vars(data_ge2, EH, CL, CW, KW, NKE, max_vif = 5)

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 object 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.
pairs_mantel

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 upper, lower and diagonal panels. Set to 'gray', 'gray', and 'gray', respectively.

pan.spacing
The space between the panels. 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)
# iris dataset
lpc <- iris %>%
group_by(Species) %>%
lpcor() %>%
pairs_mantel(names = c('setosa', 'versicolor', 'virginica'))

# mtcars dataset
mt_num <- select_numeric_cols(mtcars)
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**

```r
code

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

**Arguments**

- `.data` - The data. Must be a data frame or a grouped data passed from `group_by()`
- `resp` - The dependent variable.
- `by` - One variable (factor) to compute the function by. It is a shortcut to `group_by()`. To compute the statistics by more than one grouping variable use that function.
- `pred` - The predictor variables, set to `everything()`, 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().

plot_res  If TRUE, create a scatter plot of residual against predicted value and a normal Q-Q plot.

verbose  If verbose = TRUE then some results are shown in the console.

...  Additional arguments passed on to plot.lm

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 a grouped data passed from group_by() then the results will be returned into a list-column of data frames, containing:

Author(s)

Tiago Olivoto <tiagoolivoto@gmail.com>
perform_ammi

References


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
# Create a residual plot with 'plot_res = TRUE'
pcoeff2 <- path_coeff(data_ge2,
    resp = KW,
    pred = c(PH, EH, NKE, TKW),
    plot_res = TRUE)

# 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, block = NULL, 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 comma-separated list of unquoted variable names, i.e., `resp = c(var1, var2, var3)`, or any select helper like `resp = contains("_PLA")`.
- `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, except the error, are assumed to be fixed.**
- `verbose`: Logical argument. If `verbose = FALSE` the code will run silently.
- `...`: Arguments passed to the function `impute_missing_val()` for imputation of missing values in case of unbalanced data.

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.
- **augment**: Information about each observation in the dataset. This includes predicted values in the fitted column, residuals in the resid column, standardized residuals in the stdres column, the diagonal of the ‘hat’ matrix in the hat, and standard errors for the fitted values in the se.fit column.

Author(s)

Tiago Olivoto <tiagoolivoto@gmail.com>

References


See Also

`impute_missing_val, waas, waas_means, waasb, get_model_data`
**Examples**

```r
library(metan)
model <- performs_ammi(data_ge, ENV, GEN, REP, resp = c(GY, HM))

# PC1 x PC2 (variable GY)
p1 <- plot_scores(model)
p1

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

# Nominal yield plot (variable GY)
# Draw a convex hull polygon
plot_scores(model, type = 4)

# Unbalanced data (GEN 2 in E1 missing)
mod <-
  data_ge %>%
  remove_rows(4:6) %>%
  droplevels() %>%
  performs_ammi(ENV, GEN, REP, GY)
p2 <- plot_scores(mod)
arrange_ggplot(p1, p2, labels = c("Balanced data", "Unbalanced data"))
```

---

**Description**

Residual plots for a output model of class `anova_joint`. 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**

```r
## S3 method for class 'anova_joint'
plot(x, ...)
```

**Arguments**

- `x` An object of class `anova_joint`.
- `...` Additional arguments passed on to the function `residual_plots`
Author(s)

Tiago Olivoto <tiagoolivoto@gmail.com>

Examples

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

---

**plot.can_cor**

Plots an object of class `can_cor`

**Description**

Graphs of the Canonical Correlation Analysis

**Usage**

```r
## 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

\text{Arguments}

\begin{itemize}
\item \text{x} \quad \text{The \textit{waasb} object}
\item \text{type} \quad \text{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.}
\item \text{plot\_theme} \quad \text{The graphical theme of the plot. Default is \text{plot\_theme} = theme\_metan(). For more details, see \texttt{theme}.}
\item \text{size\_tex\_lab} \quad \text{The size of the text in axis text and labels.}
\item \text{size\_tex\_pa} \quad \text{The size of the text of the plot area. Default is 3.5.}
\item \text{x\_lab} \quad \text{The label of x-axis. Each plot has a default value. New arguments can be inserted as x\_lab = 'my label'.}
\item \text{x\_lim} \quad \text{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).}
\item \text{x\_breaks} \quad \text{The breaks to be plotted in the x-axis. Default is autumatic breaks. New arguments can be inserted as x\_breaks = c(breaks)}
\item \text{y\_lab} \quad \text{The label of y-axis. Each plot has a default value. New arguments can be inserted as y\_lab = 'my label'.}
\item \text{y\_lim} \quad \text{The range of y-axis. Default is NULL. The same arguments than x\_lim can be used.}
\item \text{y\_breaks} \quad \text{The breaks to be plotted in the x-axis. Default is autumatic breaks. The same arguments than x\_breaks can be used.}
\item \text{axis\_expand} \quad \text{Multiplication factor to expand the axis limits by to enable fitting of labels. Default is 1.1.}
\item \text{shape} \quad \text{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).}
\item \text{col\_shape} \quad \text{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").}
\item \text{col\_alpha} \quad \text{The alpha value for the color. Default is 0.9. Values must be between 0 (full transparency) to 1 (full color).}
\item \text{size\_shape} \quad \text{The size of the shape in the plot. Default is 3.5.}
\item \text{size\_bor\_tick} \quad \text{The size of tick of shape. Default is 0.3. The size of the shape will be size\_shape + size\_bor\_tick}
\item \text{labels} \quad \text{Logical arguments. If TRUE then the points in the plot will have labels.}
\item \text{main} \quad \text{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.}
\item \text{...} \quad \text{Currently not used.}
\end{itemize}

\textbf{Value}

\text{An object of class gg, ggplot.}
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 ggplot if type == "cophenetic".
Author(s)

Tiago Olivoto <tiagoolivoto@gmail.com>

Examples

```r
mean_gen <-
data_ge2 %>%
means_by(GEN) %>%
column_to_rownames("GEN")

d <- clustering(mean_gen)
plot(d, xlab = "Euclidean Distance")
```

---

**plot.corr_coef**  
*Create a correlation heat map*

Description

Create a correlation heat map for object of class `corr_coef`

Usage

```r
## 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 Correlation",
  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.
...

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

```r
## 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 automatic 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,
plot.env_dissimilarity

Plot an object of class env_dissimilarity

Description

Create dendrograms to show the dissimilarity between environments.

Usage

```r
## 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

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

### Description

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

### Usage

```r
## 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


Examples

```r
library(metan)

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

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

plot(FAI)
```

Description

Residual plots for a output model of class `gafem`. 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

```r
## S3 method for class 'gafem'
plot(x, ...)
```

Arguments

- `x` An object of class `gafem`.
- `...` Additional arguments passed on to the function `residual_plots`
Author(s)

Tiago Olivoto <tiagoolivoto@gmail.com>

Examples

```r
library(metan)
model <- gafem(data_g, GEN, REP, PH)
plot(model)
plot(model, which = c(3, 5), nrow = 2, labels = TRUE, size.lab.out = 4)
```

Description

Residual plots for a output model of class `gamem`. 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

```r
## S3 method for class 'gamem'
plot(x, var = 1, type = "res", conf = 0.95, out = "print", n.dodge = 1, check.overlap = FALSE, 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")
```
Arguments

x
  An object of class `gamem`.

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'.

n.dodge
  The number of rows that should be used to render the x labels. This is useful for displaying labels that would otherwise overlap.

check.overlap
  Silently remove overlapping labels, (recursively) prioritizing the first, last, and middle labels.

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()`.

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.

align
  Specifies whether graphs in the grid should be horizontally ("h") or vertically ("v") aligned. "hv" (default) align in both directions, "none" do not align the plot.

... Additional arguments passed on to the function `plot_grid`
Author(s)

Tiago Olivoto <tiagoolivoto@gmail.com>

Examples

```r
library(metan)
model <- gamem(data_g,
              gen = GEN,
              rep = REP,
              resp = PH)
plot(model)
```

Description

Plot an object of class `ge_cluster`

Usage

```r
## 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.

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, PH)
plot(ge_eff)
```

---

**plot.ge_factanal**  
*Plot the ge_factanal model*

**Description**

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

**Usage**

```r
## 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", "longdash", and "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`. 
plot.ge_reg

Plot an object of class ge_reg

Description

Plot the regression model generated by the function ge_reg.

Usage

```r
# 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.shape = 3,
  force.repel = 10,
  col.shape = "orange",
  col.line = "red"
)
```
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.

Value

An object of class gg,ggplot.

Author(s)

Tiago Olivoto <tiagoolivoto@gmail.com>

See Also

ge_factanal
Examples

```r
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

```r
## 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.
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. De-
defaults to 1.2
title         Logical values (Defaults to TRUE) to include automatically generated informa-
tions 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.mgidi

Plot the multi-trait genotype-ideotype distance index

Description

Makes a radar plot showing the multi-trait genotype-ideotype distance index

Usage

```r
## S3 method for class 'mgidi'
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 `mgidi`
- `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>
Examples

```r
library(metan)
model <- gamem(data_g,
gen = GEN,
rep = REP,
resp = c(KW, NR, NKE, NKR))
mgidi_index <- mgidi(model)
plot(mgidi_index)
```

**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**

```r
## S3 method for class 'mtsi'
plot(
x,
SI = 15,
radar = TRUE,
arrange.label = FALSE,
size.point = 2.5,
col.sel = "red",
col.nonsel = "black",
sizetext = 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()`. This becomes useful when the number of genotypes is large, say, more than 30.
- **arrange.label** Logical argument. If TRUE, the labels are arranged to avoid text overlapping.
- **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.
plot.performs_ammi

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


Examples

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

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, ...)

Arguments

x An object of class performs_ammi.

... Additional arguments passed on to the function residual_plots
Author(s)

Tiago Olivoto <tiagoolivoto@gmail.com>

Examples

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

Description

Plot the response surface model using a contour plot

Usage

```r
## S3 method for class 'resp_surf'
plot(
    x,
    xlab = NULL,
    ylab = NULL,
    resolution = 100,
    bins = 10,
    plot_theme = theme_metan(),
    ...
)
```

Arguments

- `x`  An object of class `resp_surf`
- `xlab, ylab`  The label for the x and y axis, respectively. Defaults to original variable names.
- `resolution`  The resolution of the contour plot. Defaults to 100. Higher values produce high-resolution plots but may increase the computation time.
- `bins`  The number of bins shown in the plot. Defaults to 10.
- `plot_theme`  The graphical theme of the plot. Default is `plot_theme = theme_metan()`. For more details, see `theme`.
- `...`  Currently not used
Value

An object of class `gg.ggplot`.

Author(s)

Tiago Olivoto <tiagoolivoto@gmail.com>

Examples

```r
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**

```r
## S3 method for class 'waas'
plot(x, ...)
```

**Arguments**

- `x` An object of class `waas`.
- `...` Additional arguments passed on to the function `residual_plots`
Author(s)

Tiago Olivoto <tiagoolivoto@gmail.com>

Examples

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

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

```r
## S3 method for class 'waasb'
plot( 
  x, 
  var = 1, 
  type = "res", 
  conf = 0.95, 
  out = "print", 
  n.dodge = 1, 
  check.overlap = FALSE, 
  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,
)```
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'.
n.dodge The number of rows that should be used to render the x labels. This is useful for display labels that would otherwise overlap.
check.overlap Silently remove overlapping labels, (recursively) prioritizing the first, last, and middle labels.
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
align Specifies whether graphs in the grid should be horizontally ("h") or vertically ("v") aligned. "hv" (default) align in both directions, "none" do not align the plot.
... Additional arguments passed on to the function plot_grid
Author(s)

Tiago Olivoto <tiagoolivoto@gmail.com>

Examples

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

Description

Plot heat maps with genotype ranking in two ways.

Usage

```r
## S3 method for class 'wsmp'
plot(x, var = 1, type = 1, y.lab = NULL, x.lab = NULL, size.lab = 12, ...)
```

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 WAASB index 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 WAASB/GY ratio.
- `y.lab`: The label of y axis. Default is 'Genotypes'.
- `x.lab`: The label of x axis. Default is 'Number of axes'.
- `size.lab`: The size of the
- `...`: 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. 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 of genotypes are shown by label colors (red) unproductive and unstable genotypes; (blue) productive, but unstable genotypes; (black) stable, but unproductive genotypes; and (green), productive and stable genotypes.

Value

An object of class gg.

Author(s)

Tiago Olivoto <tiagoolivoto@gmail.com>

Examples

library(metan)
model <- waasb(data_ge, ENV, GEN, REP, GY) %>%
  wsmp()
p1 <- plot(model)
p2 <- plot(model, type = 2)
arrange_ggplot(p1, p2, ncol = 1)

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 automatic 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.
plot_ci

Value

An object of class `ggplot`.

Author(s)

Tiago Olivoto <tiagoolivoto@gmail.com>

See Also

`plot_scores`, `plot_waasby`

Examples

```r
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

```r
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 generated 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 automatic 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 ommit 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()`.

Value

An object of class `gg, ggplot`.

Examples

```r
library(metan)
library(dplyr)

data_ge2 %>%
  select(contains('E')) %>%
  corr_ci() %>%
  plot_ci()
```
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
file.type If export = TRUE, define the type of file to be exported. Default is pdf. Graphic
file.name The name of the file for exportation, default is NULL, i.e. the files are automatical-
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.
plot_scores

Description

Plot scores of genotypes and environments in different graphical interpretations.
plot_scores

Usage

plot_scores(
  x,
  var = 1,
  type = 1,
  repel = TRUE,
  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`, `waas_means`, 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 W AASB.
- type = 4 Produces a plot with the Nominal yield x Environment PC.

**repel**
If TRUE (default), the text labels repel away from each other and away from the data points.

**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()`.

**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.
plot_scores

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. De-
fault 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 appear-
ance 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. De-
fault 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 automati-
cally 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.

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.
plot_scores

Value
An object of class gg, ggplot.

Author(s)
Tiago Olivoto <tiagoolivoto@gmail.com>

References

See Also
plot_eigen

Examples

library(metan)
# AMMI model
model <- waas(data_ge,
env = ENV,
gen = GEN,
rep = REP,
resp = everything())

# GY x PC1 for variable GY (default plot)
plot_scores(model)

# PC1 x PC2 (variable HM)
plot_scores(model,
polygon = TRUE, # Draw a convex hull polygon
var = "HM", # or var = 2 to select variable
type = 2) # type of biplot

# PC1 x PC2 (variable HM)
# Change size of plot fonts and colors
# Minimal theme
plot_scores(model,
var = "HM",
type = 2,
col.gen = "black",
col.env = "gray",
col.segm.env = "gray",
size.tex.pa = 2,
size.tex.lab = 16,
plot_theme = theme_metan_minimal())

# 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 automatic-
    ally 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 automatic 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.gamem

Predict method for gamem fits

Description

Obtains predictions from an object fitted with gamem.

Usage

## S3 method for class 'gamem'
predict(object, ...)

Arguments

object An object of class gamem
...
Currently not used

Value

A tibble with the predicted values for each variable in the model

Author(s)

Tiago Olivoto <tiagoolivoto@gmail.com>

Examples

library(metan)
model <- gamem(data_g,
gen = GEN,
rep = REP,
resp = everything())
predict(model)

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

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

Arguments

- **object**: An object of class `gge`.
- **naxis**: 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.
- **...**: Currently not used.

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


Examples

```r
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 number of axis to be used 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 example, 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 Gollob's d.f.) or Predictive success (such as cross-validation) should be used to do this. This package provides 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

```r
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

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

#### Arguments

- `object`: An object of class `waas`
- `naxis`: 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 Gollob'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))
predict

---

predict.waasb  

*Predict method for waasb fits*

Description

Obtains predictions from an object fitted with `waasb`.

Usage

```r
## S3 method for class 'waasb'
predict(object, ...)
```

Arguments

- `object`   
  An object of class `waasb`
- `...`      
  Currently not used

Value

A tibble with the predicted values for each variable in the model

Author(s)

Tiago Olivoto <tiagoolivoto@gmail.com>
print.AMMI_indexes

Examples

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

print.AMMI_indexes

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

Arguments

- **x**: An object of class AMMI_indexes.
- **which**: Which should be printed. Defaults to "stats". Other possible values are "ranks" for genotype ranking and "ssi" for the simultaneous selection index.
- **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()
predict(model)
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

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

Arguments

- `x` The Annicchiarico object
- `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

```r
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.
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

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

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

```r
## 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, 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>
print.colindiag

Examples

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

print.colindiag

Print an object of class colindiag

Description

Print the colindiag 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 'colindiag'
print(x, export = FALSE, file.name = NULL, digits = 3, ...)

Arguments

x The object of class colindiag
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)
col <- colindiag(data_ge2)
print(col)
### 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

```r
## 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

```r
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

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

Arguments

x The ecovation 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.
print.Fox

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)
library(metan)
out <- Fox(data_ge2, ENV, GEN, PH)
print(out)
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.
print.Huehn

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(model)

print.Huehn  

Print an object of class Huehn

Description

Print the Huehn 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 'Huehn'
print(x, export = FALSE, file.name = NULL, digits = 3, ...)
Arguments

x  An object of class `Huehn`.
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

```r
library(metan)
model <- Huehn(data_ge2, ENV, GEN, PH)
print(model)
```

### Description

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

Usage

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

Arguments

x  An object of class `lpcor` or `lpcor_group`.
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)
pkor <- lpcor(data_ge2, NR, NKR, NKE)
print(pkor)

# Compute the correlations for each level of the factor ENV
lpc2 <- lpcor(data_ge2,
NR, NKR, NKE,
by = ENV)
print(lpc2)

print.mgidi

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

Description

Print an object of class mgidi Print a mgidi 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 'mgidi'
print(x, export = FALSE, file.name = NULL, digits = 4, ...)

Arguments

x An object of class mgidi.
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>
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 'mtsii'
print(x, export = FALSE, file.name = NULL, digits = 4, ...)

Arguments

x An object of class mtsi.
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)
# Based on stability only
MTSI_MODEL <- waasb(data_ge,
  resp = c(GY, HM),
  gen = GEN,
  env = ENV,
  rep = REP
print(mtsi(MTSI_MODEL))
print.path.coeff

print(path_coeff)

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

```r
## 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

```r
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**

```r
## 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**

```r
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.
print.Shukla

Usage

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

Arguments

- `x`: The Shukla object.
- `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

```r
library(metan)
Sch <- Schmildt(data_ge2,
               env = ENV,
               gen = GEN,
               rep = REP,
               resp = PH)
print(Sch)
```

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

```r
## 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

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, PH)
print(model)

print.Thennarasu

Print an object of class Thennarasu

Description

Print the Thennarasu 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 'Thennarasu'
print(x, export = FALSE, file.name = NULL, digits = 3, ...)

Arguments

x An object of class Thennarasu.
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 <- Thennarasu(data_ge2, ENV, GEN, 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

```r
## 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

```r
library(metan)
model <- waas(data_ge, 
    resp = c(GY, HM), 
    gen = GEN, 
    env = ENV, 
    rep = REP
)
print(model)
```
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, a *.txt file is exported to the working directory

blup A logical argument. If TRUE, 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 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**

```r
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**

```r
rbind_fill(..., fill = ".")
```

**Arguments**

- `...`: Input dataframes.
- `fill`: What to fill? Default is ".".

**Value**

A data frame.

**Author(s)**

Tiago Olivoto <tiagoolivoto@gmail.com>

**Examples**

```r
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**

```r
reorder_cormat(x)
```
**Arguments**

- **x** The correlation matrix

**Value**

The ordered correlation matrix

**Author(s)**

Tiago Olivoto <tiagoolivoto@gmail.com>

**Examples**

```r
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**

```r
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:

\[ R_{v_i} = \frac{(N_{\text{max}} - N_{\text{min}})}{(O_{\text{max}} - O_{\text{min}})} \times (O_i - O_{\text{max}}) + N_{\text{max}} \]

Where \( R_{v_i} \) is the rescaled value of the \( i \)th position of the variable/ vector; \( N_{\text{max}} \) and \( N_{\text{min}} \) are the new maximum and minimum values; \( O_{\text{max}} \) and \( O_{\text{min}} \) are the maximum and minimum values of the original data, and \( O_i \) is the \( i \)th 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 \( \text{.data} \) argument and selecting one or more variables to be scaled using \( \ldots \). The function will return the original variables in \( \text{.data} \) plus the rescaled variable(s) with the prefix \_res. By using the function \text{group_by} from \text{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 \text{values}. The output, of course, will be a numeric vector of rescaled values.

Value

A numeric vector if \text{values} is used as input data or a tibble if a data frame is used as input in \text{.data}.

Author(s)

Tiago Olivoto <tiagoolivoto@gmail.com>

Examples

```r
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' within ENV;
library(dplyr)
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 $i$th genotype in the $j$th 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>
residual_plots

References


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()
```

residual_plots

Several types of residual plots

Description

Residual plots for a output model of class `performs_ammi`, `waas`, `anova_ind`, and `anova_joint`. 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
residual_plots(
  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,
  align = "hv",
  ...
)

Arguments

x | An object of class performs_ammi, waas, anova_joint, or gafem
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 panel. Defaults to `NULL`

**align**  
Specifies whether graphs in the grid should be horizontally (`"h"`) or vertically (`"v"`) aligned. "hv" (default) align in both directions, "none" do not align the plot.

**...**  
Additional arguments passed on to the function `plot_grid`

**Author(s)**

Tiago Olivoto <tiagoolivoto@gmail.com>

**Examples**

```r
library(metan)
model <- performs_ammi(data_ge, ENV, GEN, REP, GY)

# Default plot
plot(model)

# Normal Q-Q plot
# Label possible outliers
plot(model,
    which = 2,
    labels = TRUE)

# Residual vs fitted,
# Normal Q-Q plot
# Histogram of raw residuals
# All in one row
plot(model,
    which = c(1, 2, 5),
    nrow = 1)
```

---

**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

```r
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

```r
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)
```
Description


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


Select_helper

See Also

superiority, ecovalence, ge_stats, Annicchiarico

Examples

library(metan)
Sch <- Schmildt(data_ge2,
   env = ENV,
   gen = GEN,
   rep = REP,
   resp = PH)
print(Sch)

Usage

difference_var(prefix, suffix)

intersect_var(prefix, suffix)

union_var(prefix, suffix)

width_of(n, vars = peek_vars(fn = "width_of"))
width_greater_than(n, vars = peek_vars(fn = "width_greater_than"))

width_less_than(n, vars = peek_vars(fn = "width_less_than"))

lower_case_only(vars = peek_vars(fn = "lower_case_only"))

upper_case_only(vars = peek_vars(fn = "upper_case_only"))

title_case_only(vars = peek_vars(fn = "title_case_only"))

Arguments

prefix A prefix that start the variable name.
suffix A suffix that end the variable name.
n The length of variable names to select. For width_of() the selected variables contains \(n\) characters. For width_greater_than() and width_less_than() the selected variables contains greater and less characters than \(n\), respectively.
vars A character vector of variable names. When called from inside selecting functions like select_cols these are automatically set to the names of the table.

Examples

library(metan)

# Select variables that start with "C" and not end with "D".
data_ge2 %>%
select_cols(difference_var("C", "D"))

# Select variables that start with "C" and end with "D".
data_ge2 %>%
select_cols(intersect_var("C", "D"))

# Select variables that start with "C" or end with "D".
data_ge2 %>%
select_cols(union_var("C", "D"))

# Select variables with width name of 4
data_ge2 %>%
select_cols(width_of(4))

# Select variables with width name greater than 2
data_ge2 %>%
select_cols(width_greater_than(2))

# Select variables with width name less than 3
data_ge2 %>%
select_cols(width_less_than(3))
# Creating data with messy column names
df <- head(data_ge, 3)
colnames(df) <- c("Env", "gen", "Rep", "GY", "hm")
select_cols(df, lower_case_only())
select_cols(df, upper_case_only())
select_cols(df, title_case_only())

---

**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 incorporate the mean performance and stability into a single selection criteria.

**Usage**

```r
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 `ShuklaVar`.
- **ssiShukaVar**  
The simultaneous selection index (`ssiShukaVar = rMean + rShukaVar`).
solve_svd

Author(s)

Tiago Olivoto <tiagoolivoto@gmail.com>

References


Examples

```r
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

```r
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

```r
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)
as.split_factors(.data, keep_factors = FALSE)
is.split_factors(x)
```

#### 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 split the data.
- `keep_factors`: Should the grouping columns be kept?
- `x`: An object to check for class `split_factors`.

#### Details
This function is used to split a data frame into a named list where each element is a level of the grouping variable (or combination of grouping variables).

- `split_factors()` Split a data frame by factors.
- `as.split_factors()` coerce to an object of class `split_factors`
- `is.split_factors()` check if an object is of class `split_factors`

#### 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)

spdata <- as.split_factors(iris)
is.split_factors(spdata)

stars_pval(p_value)

Arguments

p_value A numeric vector of p-values

Details

Mapping from p_value ranges to symbols:

- 0 - 0.0001: '****'
- 0.0001 - 0.001: '***'
- 0.001 - 0.01: '**'
- 0.01 - 0.05: '*'
- 0.05 - 1.0: 'ns'

Value

A character vector containing the same number of elements as p-value, with an attribute "legend" providing the conversion pattern.
Author(s)
Tiago Olivoto <tiagoolivoto@gmail.com>

Examples

```r
p_vals <- c(0.01, 0.043, 0.1, 0.0023, 0.000012)
stars_pval(p_vals)
```

---

**superiority**  
*Lin e Binns’ superiority index*

Description

Nonparametric stability analysis using the superiority index proposed by Lin & Binns (1988).

Usage

```r
superiority(.data, env, gen, 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.
- `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 (Pi_a), favorable (Pi_f) and unfavorable (Pi_u) environments.

Author(s)

Tiago Olivoto, <tiagoolivoto@gmail.com>
References


See Also

Annicchiarico, ecovalence, ge_stats

Examples

library(metan)
out <- superiority(data_ge2, ENV, GEN, PH)
print(out)

themes

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"
- `alpha_color()`: Return a semi-transparent color based on a color name and an alpha value. For more details see `colors`.

Usage

`theme_metan(grid = "none", col.grid = "white", color.background = "gray95")`

`theme_metan_minimal()`

`transparent_color()`

`alpha_color(color, alpha = 50)`

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
Thennarasu

color.background
The color for the panel background.

color
A color name.

alpha
An alpha value for transparency (0 < alpha < 1).

Author(s)
Tiago Olivoto <tiagoolivoto@gmail.com>

---

Thennarasu  Thennarasu’s stability statistics

Description

Usage
Thennarasu(.data, env, gen, resp, verbose = TRUE)

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

e 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 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
Examples

```r
library(metan)
out <- Thennarasu(data_ge, ENV, GEN, GY)
print(out)
```

---

**to_factor**

*Encode variables to a factor*

**Description**

Function to quick mutate columns to factor.

**Usage**

```r
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**

```r
library(metan)
PH_EH_to_factor <- to_factor(data_ge2, PH, EH)
PH_EH_to_factor <- to_factor(data_ge2, 4:5)
```
tukey_hsd

Tukey Honest Significant Differences

Description

Helper function to perform Tukey post-hoc tests. It is used in gafem.

Usage

tukey_hsd(model, ..., out = "long")

Arguments

model  an object of class aov or lm.

...  other arguments passed to the function TukeyHSD(). These include:

• which: A character vector listing terms in the fitted model for which the intervals should be calculated. Defaults to all the terms.

• ordered: A logical value indicating if the levels of the factor should be ordered according to increasing average in the sample before taking differences. If ordered is true then the calculated differences in the means will all be positive. The significant differences will be those for which the lwr end point is positive.

out  The format of outputs. If out = "long" a 'long' format (tibble) is returned. If out = "wide", a matrix with the adjusted p-values for each term is returned.

Value

A tibble data frame containing the results of the pairwise comparisons (if out = "long") or a "list-columns" with p-values for each term (if out = "wide").

Examples

library(metan)
mod <- lm(PH ~ GEN + REP, data = data_g)
tukey_hsd(mod)
tukey_hsd(mod, out = "wide")
Utilities for handling with classes

Description
Utilities for handling with classes

Usage
add_class(x, class)
has_class(x, class)
remove_class(x, class)
set_class(x, class)

Arguments
x      An object
class  The class to add or remove

Details
• add_class(): add a class to the object x keeping all the other class(es).
• has_class(): Check if a class exists in object x and returns a logical value.
• set_class(): set a class to the object x.
• remove_class(): remove a class from the object x.

Value
The object x with the class added or removed.

Author(s)
Tiago Olivoto <tiagoolivoto@gmail.com>

Examples
library(metan)
df <-
data_ge2 %>%
add_class("my_class")
class(df)
has_class(df, "my_class")
remove_class(df, "my_class") %>% class()
set_class(df, "data_frame") %>% class()

---

**utils_mat**

*Utilities for handling with matrices*

### Description

These functions help users to make upper, lower, or symmetric matrices easily.

### Usage

```r
make_upper_tri(x, diag = NA)
make_lower_tri(x, diag = NA)
make_sym(x, make = "upper", diag = NA)
tidy_sym(x, keep_diag = TRUE)
```

### Arguments

- `x`: A matrix to apply the function. It must be a symmetric (square) matrix in `make_upper_tri()` and `make_lower_tri()` or a triangular matrix in `make_sym()`. `tidy_sym()` accepts both symmetrical or triangular matrices.
- `diag`: What show in the diagonal of the matrix. Default to `NA`.
- `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.
- `keep_diag`: Keep diagonal values in the tidy data frame? Defaults to `TRUE`.

### Details

- `make_upper_tri()` makes an upper triangular matrix using a symmetric matrix.
- `make_lower_tri()` makes a lower triangular matrix using a symmetric matrix.
- `make_sym()` makes a lower triangular matrix using a symmetric matrix.
- `tidy_sym()` transform a symmetric matrix into tidy data frame.

### Value

An upper, lower, or symmetric matrix, or a tidy data frame.

### Author(s)

Tiago Olivoto <tiagoolivoto@gmail.com>
Examples

```r
library(metan)
m <- cor(select_cols(data_ge2, 5:10))
make_upper_tri(m)
make_lower_tri(m)
make_lower_tri(m) %>%
  make_sym(diag = 0)
tidy_sym(m)
tidy_sym(make_lower_tri(m))
```

---

**utils_na**

Utilities for handling with NA values

**Description**

- `has_na()`: Check for NA values in the data and return a logical value.
- `random_na()`: Generate random NA values in a two-way table based on a desired proportion.
- `remove_cols_na()`: Remove columns with NA values.
- `remove_rows_na()`: Remove rows with NA values.
- `select_cols_na()`: Select columns with NA values.
- `select_rows_na()`: Select rows with NA values.
- `replace_na()`: Replace missing values

**Usage**

```r
remove_rows_na(.data, verbose = TRUE)
remove_cols_na(.data, verbose = TRUE)
select_cols_na(.data, verbose = TRUE)
select_rows_na(.data, verbose = TRUE)
has_na(.data)
replace_na(.data, ..., replace = 0)
random_na(.data, prop)
```
Arguments

- `.data` (A data frame or tibble)
- `verbose` (Logical argument. If TRUE (default) shows in console the rows or columns deleted.)
- `...` (Variables to replace NAs. If ... is null then all variables with NA will be replaced. It must be a single variable name or a comma-separated list of unquoted variables names. Select helpers are also allowed.)
- `replace` (The value used for replacement. Defaults to 0. Use replace = "colmeans" to replace missing values with column means.)
- `prop` (The proportion (percentage) of NA values to generate in .data.)

Value

A data frame with rows or columns with NA values deleted.

Examples

```r
library(metan)
data_with_na <- data_g
data_with_na[c(1, 5, 10), c(3:5, 10:15)] <- NA
data_with_na
has_na(data_with_na)
remove_cols_na(data_with_na)
remove_rows_na(data_with_na)
select_cols_na(data_with_na)
select_rows_na(data_with_na)
replace_na(data_with_na)
```

Description

- `all_lower_case()`: Translate all non-numeric strings of a data frame to lower case ("Env" to "env").
- `all_upper_case()`: Translate all non-numeric strings of a data frame to upper case (e.g., "Env" to "ENV").
- `all_title_case()`: Translate all non-numeric strings of a data frame to title case (e.g., "ENV" to "Env").
- `extract_number()`: Extract the number(s) of a string.
- `extract_string()`: Extract all strings, ignoring case.
- `find_text_in_num()`: Find text characters in a numeric sequence and return the row index.
• has_text_in_num(): Inspect columns looking for text in numeric sequence and return a warning if text is found.
• remove_space(): Remove all blank spaces of a string.
• 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.
• tidy_strings(): Tidy up characters strings, non-numeric columns, or any selected columns in a data frame by putting all word in upper case, replacing any space, tabulation, punctuation characters by '_', and putting '_' between lower and upper case. Suppose that str = c("Env1", "env 1", "env.1") (which by definition should represent a unique level in plant breeding trials, e.g., environment 1) is subjected to tidy_strings(str): the result will be then c("ENV_1", "ENV_1", "ENV_1"). See Examples section for more examples.

Usage

all_upper_case(.data, ...)

all_lower_case(.data, ...)

all_title_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
)

find_text_in_num(.data, ...)

has_text_in_num(.data)

remove_space(.data, ...)
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 = "",
  ignore_case = FALSE,
  drop = FALSE,
  pull = FALSE,
  .before = NULL,
  .after = NULL
)

round_cols(.data, ..., digits = 2)

tidy_strings(.data, ..., sep = ",")

Arguments

.data A data frame

... The argument depends on the function used.
  
  • For round_cols() ... are the variables to round. If no variable is in-
    formed, all the numeric variables from .data are used.
  
  • For all_lower_case(), all_upper_case(), all_title_case(), remove_strings(),
    and tidy_strings() ... are the variables to apply the function. If no vari-
    able is informed, the function will be applied to all non-numeric variables
    in .data.

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 exist-
  ing 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_string(), replace_number(), extract_string(), and extract_number() one-based column index or column name where to add the new columns.

pattern A string to be matched. Regular Expression Syntax is also allowed.
replacement A string for replacement.
ignore_case If FALSE (default), the pattern matching is case sensitive and if TRUE, case is ignored during matching.

digits The number of significant figures.
sep A character string to separate the terms. Defaults to "_".

Author(s)
Tiago Olivoto <tiagoolivoto@gmail.com>

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)

############ Find text in numeric sequences ###########
mixed_text <- data.frame(data_ge)
mixed_text[2, 4] <- "2..503"
mixed_text[3, 4] <- "3.2075"
find_text_in_num(mixed_text, GY)

############# upper, lower and title cases ############
gen_text <- c("GEN 1", "Gen 1", "gen 1")
all_lower_case(gen_text)
all_upper_case(gen_text)
all_title_case(gen_text)

# A whole data frame
all_lower_case(data_ge)

############### Tidy up messy text string ###############
messy_env <- c("ENV 1", "Env 1", "Env1", "env1", "Env.1", "Env_1")
tidy_strings(messy_env)
messy_gen <- c("GEN1", "gen 2", "Gen.3", "gen-4", "Gen_5", "GEN_6")
tidy_strings(messy_gen)
messy_int <- c("EnvGen", "Env_Gen", "env gen", "Env Gen", "ENV.GEN", "ENV_GEN")
tidy_strings(messy_int)

library(tibble)
# Or a whole data frame
df <- tibble(Env = messy_env,
gen = messy_gen,
Env_GEN = interaction(Env, gen),
y = rnorm(6, 300, 10))
df
tidy_strings(df)
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.

- **colnames_to_lower()**: Translate all column names to lower case.

- **colnames_to_upper()**: Translate all column names to upper case.

- **colnames_to_title()**: Translate all column names to title case.

- **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. If pull = TRUE then the concatenated variable is pull out to a vector. This is specially useful when using concatenate to add columns to a data frame with add_cols().

- **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)

colnames_to_lower(.data)

colnames_to_upper(.data)

colnames_to_title(.data)

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

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 nth variable from the end (for select_last_col()) or from the begin (for select_first_col())

Author(s)

Tiago Olivoto <tiagoolivoto@gmail.com>

Examples

library(metan)
### Adding columns

#### Variables x and y .after last column
```r
data_ge %>%
  add_cols(x = 10,
           y = 30)
```

#### Variables x and y .before the variable GEN
```r
data_ge %>%
  add_cols(x = 10,
           y = 30,
           .before = "GEN")
```

#### Creating a new variable based on the existing ones.
```r
data_ge %>%
  add_cols(GY2 = GY^2,
           GY2_HM = GY2 + HM,
           .after = "GY")
```

### Reordering columns
```r
reorder_cols(data_ge2, NKR, .before = "ENV")
reorder_cols(data_ge2, ENV, GEN, .after = "ED")
```

#### Selecting and removing columns
```r
select_cols(data_ge2, GEN, REP)
remove_cols(data_ge2, GEN, REP)
```

#### Selecting and removing rows
```r
select_rows(data_ge2, 2:3)
remove_rows(data_ge2, 2:3)
```

#### Concatenating columns
```r
concatenate(data_ge, ENV, GEN, REP)
concatenate(data_ge, ENV, GEN, REP, drop = TRUE)
```

# Combine with add_cols() and replace_string()
```r
data_ge2 %>
  add_cols(ENV_GEN = concatenate(., ENV, GEN, pull = TRUE),
            .after = "GEN") %>%
  replace_string(ENV_GEN,
                 pattern = "H",
                 replacement = "HYB_",
                 .after = "ENV_GEN")
```

#### Formatting column names
```r
# Creating data with messy column names
df <- head(data_ge, 3)
colnames(df) <- c("Env", "gen", "Rep", "GY", "hm")
df
colnames_to_lower(df)
colnames_to_upper(df)
colnames_to_title(df)
```

### Adding rows
```r
data_ge %>%
  add_rows(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)
```

---

**utils_stats**

*Useful functions for computing descriptive statistics*

**Description**

- **The following functions compute descriptive statistics by levels of a factor or combination of factors quickly.**
  - `cv_by()` For computing coefficient of variation.
  - `max_by()` For computing maximum values.
  - `means_by()` For computing arithmetic means.
  - `min_by()` For computing minimum values.
  - `n_by()` For getting the length.
  - `sd_by()` For computing sample standard deviation.
  - `sem_by()` For computing standard error of the mean.

- **Useful functions for descriptive statistics. All of them work naturally with %>%%, handle grouped data and multiple variables (all numeric variables from .data by default).**
  - `av_dev()` computes the average absolute deviation.
  - `ci_mean()` computes the confidence interval for the mean.
  - `cv()` computes the coefficient of variation.
  - `freq_table()` Computes frequency table. Handles grouped data.

- **hmean(),gmean() computes the harmonic and geometric means, respectively.** The harmonic mean is the reciprocal of the arithmetic mean of the reciprocals. The geometric mean is the \( n \)th root of \( n \) products.
  - `kurt()` computes the kurtosis like used in SAS and SPSS.
  - `range_data()` Computes the range of the values.
- `sd_amo()`, `sd_pop()` Computes sample and populational standard deviation, respectively.
- `sem()` computes the standard error of the mean.
- `skew()` computes the skewness like used in SAS and SPSS.
- `sum_dev()` computes the sum of the absolute deviations.
- `sum_sq_dev()` computes the sum of the squared deviations.
- `var_amo()`, `var_pop()` computes sample and populational variance.
- `valid_n()` Return the valid (not NA) length of a data.

`desc_stat` is wrapper function around the above ones and can be used to compute quickly all these statistics at once.

**Usage**

```r
av_dev(.data, ..., na.rm = FALSE)

ci_mean(.data, ..., na.rm = FALSE, level = 0.95)

cv(.data, ..., na.rm = FALSE)

freq_table(.data, ...)

hmean(.data, ..., na.rm = FALSE)

gmean(.data, ..., na.rm = FALSE)

kurt(.data, ..., na.rm = FALSE)

range_data(.data, ..., na.rm = FALSE)

sd_amo(.data, ..., na.rm = FALSE)

sd_pop(.data, ..., na.rm = FALSE)

sem(.data, ..., na.rm = FALSE)

skew(.data, ..., na.rm = FALSE)

sum_dev(.data, ..., na.rm = FALSE)

sum_sq_dev(.data, ..., na.rm = FALSE)

var_pop(.data, ..., na.rm = FALSE)

var_amo(.data, ..., na.rm = FALSE)

valid_n(.data, ..., na.rm = FALSE)

cv_by(.data, ..., na.rm = FALSE)
```
max_by(.data, ..., na.rm = FALSE)
means_by(.data, ..., na.rm = FALSE)
min_by(.data, ..., na.rm = FALSE)
n_by(.data, ..., na.rm = FALSE)
sd_by(.data, ..., na.rm = FALSE)
sem_by(.data, ..., na.rm = FALSE)

Arguments

.data A data frame or a numeric vector.

... The argument depends on the function used.

• For *_by functions, ... is one or more categorical variables for grouping the data. Then the statistic required will be computed for all numeric variables in the data. If no variables are informed in ..., the statistic will be computed ignoring all non-numeric variables in .data.

• For the other statistics, ... is a comma-separated of unquoted variable names to compute the statistics. If no variables are informed in n ..., the statistic will be computed for all numeric variables in .data.

na.rm A logical value indicating whether NA values should be stripped before the computation proceeds. Defaults to FALSE.

level The confidence level for the confidence interval of the mean. Defaults to 0.95.

Value

• Functions *_by() returns a tbl_df with the computed statistics by each level of the factor(s) declared in ....

• All other functions return a named integer if the input is a data frame or a numeric value if the input is a numeric vector.

Author(s)

Tiago Olivoto <tiagoolivoto@gmail.com>

Examples

library(metan)
# means of all numeric variables by ENV
means_by(data_ge2, GEN, ENV)

# Coefficient of variation for all numeric variables
# by GEN and ENV
cv_by(data_ge2, GEN, ENV)

# Skewness of a numeric vector
set.seed(1)
nvec <- rnorm(200, 10, 1)
skew(nvec)

# Confidence interval 0.95 for the mean
# All numeric variables
# Grouped by levels of ENV
data_ge2 %>%
  group_by(ENV) %>%
  ci_mean()

# standard error of the mean
# Variable PH and EH
sem(data_ge2, PH, EH)

# Frequency table for variable NR
data_ge2 %>%
  freq_table(NR)

---

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,
  block = NULL,
  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)`.

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, except the error, are assumed to be fixed.

mresp The new maximum value after rescaling the response variable. By default, all variables in `resp` are rescaled so that the maximum value is 100 and the minimum value is 0 (i.e., `mresp = 100`). It must be a numeric vector of the same length of `resp` if rescaling is assumed to be different across variables, e.g., if for the first variable smaller values are better and for the second one, higher values are better, then `mresp = c(0,100)` must be used. Numeric value of length 1 will be recycled with a warning message.

wresp The weight for the response variable(s) for computing the WAASBY index. By default, all variables in `resp` have equal weights for mean performance and stability (i.e., `wresp = 50`). It must be a numeric vector of the same length of `resp` to assign different weights across variables, e.g., if for the first variable equal weights for mean performance and stability are assumed and for the second one, a higher weight for mean performance (e.g. 65) is assumed, then `wresp = c(50,65)` must be used. Numeric value of length 1 will be recycled with a warning message.

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:

\[ W\text{AAS}_i = \frac{1}{\sum_{k=1}^{P} |IPCA_{ik} \times EP_k|/ \sum_{k=1}^{P} EP_k} \]

where \( W\text{AAS}_i \) is the weighted average of absolute scores of the \( i \)th genotype; \( IPCA_{ik} \) 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 \( \text{prob} = 0.05 \), all axis that are significant considering this probability level are used. The number of axis can be also informed by declaring \( \text{naxis} = x \). 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. \( \text{WgtResponse} \), the weight for the response variable in estimating WAASB, \( \text{WgtWAAS} \) the weight for stability, \( \text{Ngen} \) the number of genotypes, \( \text{Nenv} \) the number of environments, \( \text{OVmean} \) the overall mean, \( \text{Min} \) the minimum observed (returning the genotype and environment), \( \text{Max} \) the maximum observed, \( \text{MinENV} \) the environment with the lower mean, \( \text{MaxENV} \) the environment with the larger mean observed, \( \text{MinGEN} \) the genotype with the lower mean, \( \text{MaxGEN} \) the genotype with the larger.
- **augment**: Information about each observation in the dataset. This includes predicted values in the fitted column, residuals in the resid column, standardized residuals in the stdres column, the diagonal of the ‘hat’ matrix in the hat, and standard errors for the fitted values in the se.fit column.
- **probint** The p-value for the genotype-vs-environment interaction.

**Author(s)**

Tiago Olivoto <tiagoolivoto@gmail.com>

**References**


**See Also**

`waas_means waasb get_model_data`
Examples

library(metan)

# Example 1: Analyzing all numeric variables considering p-value <= 0.05 to compute the WAAS.
model <- waas(data_ge,
               env = ENV,
               gen = GEN,
               rep = REP,
               resp = everything())

# Residual plot (first variable)
plot(model)

# Get the WAAS index
get_model_data(model, "WAAS")

# Plot WAAS and response variable
plot_scores(model, type = 3)

# Example 2: 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 = everything(),
               naxis = 1, # Only to compare with PC1
               wresp = 60)

# Get the WAAS index (it will be |PC1|)
get_model_data(model2)

# Get values for IPCA1
get_model_data(model2, "PC1")

# Example 3: Analyzing GY and HM assuming a random-effect model. Smaller values for HM and higher values for GY are better.
model3 <- waas(data_ge,
               env = ENV,
               gen = GEN,
rep = REP,
resp = c(GY, HM),
mresp = c(100, 0),
wresp = c(60, 40))

# Get the ranks for the WAASY index
get_model_data(model3, what = "OrWAASY")

---

**waasb**  
*Weighted Average of Absolute Scores*

**Description**

Compute the Weighted Average of Absolute Scores (Olivoto et al., 2019) for quantifying the stability of *g* genotypes conducted in *e* environments using linear mixed-effect models.

**Usage**

```r
waasb(
  .data,
  env,
  gen,
  rep,
  resp,
  block = NULL,
  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).`
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).

The new maximum value after rescaling the response variable. By default, all variables in resp are rescaled so that the maximum value is 100 and the minimum value is 0 (i.e., mresp = 100). It must be a numeric vector of the same length of resp if rescaling is assumed to be different across variables, e.g., if for the first variable smaller values are better and for the second one, higher values are better, then mresp = c(0,100) must be used. Numeric value of length 1 will be recycled with a warning message.

The weight for the response variable(s) for computing the WAASBY index. By default, all variables in resp have equal weights for mean performance and stability (i.e., wresp = 50). It must be a numeric vector of the same length of resp to assign different weights across variables, e.g., if for the first variable equal weights for mean performance and stability are assumed and for the second one, a higher weight for mean performance (e.g. 65) is assumed, then wresp = c(50,65) must be used. Numeric value of length 1 will be recycled with a warning message.

The effects of the model assumed to be random. Defaults to random = "gen". See Details to see the random effects assumed depending on the experimental design of the trials.

The probability for estimating confidence interval for BLUP’s prediction.

Logical argument set to TRUE. If FALSE the within-environment ANOVA is not performed.

Logical argument. If verbose = FALSE the code will run silently.

Arguments passed to the function impute_missing_val() for imputation of missing values in the matrix of BLUPs for genotype-environment interaction, thus allowing the computation of the WAASB index.

### Details

The weighted average of absolute scores is computed considering all Interaction Principal Component Axis (IPCA) from the Singular Value Decomposition (SVD) of the matrix of genotype-environment interaction (GEI) effects generated by a linear mixed-effect model, as follows:

$$ WAASB_i = \frac{\sum_{k=1}^{p} |IPCA_{ik} \times EP_k|}{\sum_{k=1}^{p} EP_k} $$

where $WAASB_i$ is the weighted average of absolute scores of the $i$th genotype; $IPCA_{ik}$ is the score of the $i$th genotype in the $k$th Interaction Principal Component Axis (IPCA); and $EP_k$ is the explained variance of the $k$th IPCA for $k = 1,2,...,p$, considering $p = \min(g - 1; e - 1)$.

The nature of the effects in the model is chosen with the argument random. By default, the experimental design considered in each environment is a randomized complete block design. If block is informed, a resolvable alpha-lattice design (Patterson and Williams, 1976) is implemented. The following six models can be fitted depending on the values of random and block arguments.
• **Model 1**: block = NULL and random = "gen" (The default option). This model considers a Randomized Complete Block Design in each environment assuming genotype and genotype-environment interaction as random effects. Environments and blocks nested within environments are assumed to fixed factors.

• **Model 2**: block = NULL and random = "env". This model considers a Randomized Complete Block Design in each environment treating environment, genotype-environment interaction, and blocks nested 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 in each environment assuming a random-effect model, i.e., all effects (genotypes, environments, genotype-vs-environment interaction and blocks nested within environments) are assumed to be random factors.

• **Model 4**: block is not NULL and random = "gen". This model considers an alpha-lattice design in each environment assuming genotype, genotype-environment interaction, and incomplete blocks nested within complete replicates as random to make use of inter-block information (Mohring et al., 2015). Complete replicates nested within environments and environments are assumed to be fixed factors.

• **Model 5**: block is not NULL and random = "env". This model considers an alpha-lattice design in each environment assuming genotype as fixed. All other sources of variation (environment, genotype-environment interaction, complete replicates nested within environments, and incomplete blocks nested within replicates) are assumed to be random factors.

• **Model 6**: block is not NULL and random = "all". This model considers an alpha-lattice design in each environment assuming all effects, except the intercept, as random factors.

**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 tibble with the response variable, the scores of all IPCAs, the estimates of Weighted Average of Absolute Scores, and WAASBY (the index that considers the weights for stability and mean performance in the genotype ranking), and their respective ranks.
- **BLUPgen** The random effects and estimated BLUPS for genotypes (If random = "gen" or random = "all")
- **BLUPenv** The random effects and estimated BLUPS for environments, (If random = "env" or random = "all")
- **BLUPint** The random effects and estimated BLUPS of all genotypes in all environments.
- **PCA** The results of Principal Component Analysis with the eigenvalues and explained variance of the matrix of genotype-environment effects estimated by the linear fixed-effect model.
- **MeansGxE** The phenotypic means of genotypes in the environments.
• **Details** A list summarizing the results. The following information are shown: Nenv, the number of environments in the analysis; Ngen the number of genotypes in the analysis; mresp The value attributed to the highest value of the response variable after rescaling it; wresp The weight of the response variable for estimating the WAASBY index. Mean the grand mean; SE the standard error of the mean; SD the standard deviation. CV the coefficient of variation of the phenotypic means, estimating WAASB, Min the minimum value observed (returning the genotype and environment), Max the maximum value observed (returning the genotype and environment); 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 tibble with the genetic parameters (if random = "gen" or random = "all") with the following columns: Phenotypic variance the phenotypic variance; Heritability the broad-sense heritability; GEr2 the coefficient of determination of the interaction effects; Heritability of means the heritability on the mean basis; Accuracy the selective accuracy; rge the genotype-environment correlation; CVg the genotypic coefficient of variation; CVr the residual coefficient of variation; CV ratio the ratio between genotypic and residual coefficient of variation.

• **residuals** The residuals of the model.

**Author(s)**
Tiago Olivoto <tiagoolivoto@gmail.com>

**References**


**See Also**
mts, waas, get_model_data, plot_scores

**Examples**

```r
library(metan)
#===============================================================#
# Example 1: Analyzing all numeric variables assuming genotypes #
# as random effects with equal weights for mean performance and #
# stability #
#===============================================================#
model <- waasb(data_ge,
    env = ENV,
    gen = GEN,
)```
rep = REP,
resp = everything()
# Distribution of random effects (first variable)
plot(model, type = "re")

# Genetic parameters
get_model_data(model, "genpar")

# Example 2: Analyzing variables that starts with "N" #
# assuming environment as random effects with higher weight for #
# response variable (65) for the three traits. #
#===============================================================#
model2 <- waasb(data_ge2,
                 env = ENV,
                 gen = GEN,
                 rep = REP,
                 random = "env",
                 resp = starts_with("N"),
                 wresp = 65)

# Get the index WAASBY
get_model_data(model2, what = "WAASBY")

# Plot the scores (response x WAASB)
plot_scores(model2, type = 3)

# Example 3: Analyzing GY and HM assuming a random-effect model. #
# Smaller values for HM and higher values for GY are better. #
# To estimate WAASBY, higher weight for the GY (60%) and lower #
# weight for HM (40%) are considered for mean performance. #
#===============================================================#
model3 <- waasb(data_ge,
                 env = ENV,
                 gen = GEN,
                 rep = REP,
                 resp = c(GY, HM),
                 random = "all",
                 mresp = c(100, 0),
                 wresp = c(60, 40))

# Get P-values for Likelihood-ratio test
get_model_data(model3, "pval_lrt")

# Get the random effects
get_model_data(model3, what = "ranef")
waas_means

# Get the ranks for the WAASB index
get_model_data(model3, what = "OrWAASB")

## 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**

```r
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 the 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 Component Axis are iteractively retained up to the explained variance (eigenvalues in the singular value decomposition of the matrix with the interaction effects)
be greater than or equal to \(\min_{\text{expl\_var}}\). 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 \(\text{verbose} = \text{FALSE}\) the code is run silently.

Arguments passed to the function \texttt{impute\_missing\_val()} for imputation of missing values in case of unbalanced data.

**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 \(k = 1,2,\ldots,p\), where \(p\) is the number of IPCAs that explain at least an amount of the genotype-interaction variance declared in the argument \(\min_{\text{expl\_var}}\).

**Value**

An object of class \texttt{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\_means** A tbl\_df containing the genotype-environment means.
- **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 \(<\text{tiagoolivoto@gmail.com}>\)

**References**


**See Also**

\texttt{waas waasb}
**Examples**

```r
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 = "OrWAAS")
get_model_data(model2, what = "OrWAAS")
```

---

**wmsp**

*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**

```r
wmsp(
  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 \( m_{\text{resp}} = 0 \) 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. See 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` must be multiple of `increment`. If this assumption is not valid, an error will be occur.

**prob**
The p-value for considering an interaction principal component axis significant. For details see the [Details](#) section.

**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


### See Also

- `resca`
Examples

```r
library(metan)
model <- waasb(data_ge2,
    env = ENV,
    gen = GEN,
    rep = REP,
    resp = PH)
scenarios <- wsm(model)
```
Index

*Topic data
  data_alpha, 43
  data_g, 44
  data_ge, 45
  data_ge2, 46
  int.effects, 93
  meansGxE, 103

add_class (utils_class), 205
add_cols (utils_rows_cols), 213
add_rows (utils_rows_cols), 213
all_lower_case (utils_num_str), 208
all_pairs (utils_rows_cols), 213
all_title_case (utils_num_str), 208
all_upper_case (utils_num_str), 208
alpha_color (themes), 201
AMMI_indexes, 6, 66, 71, 82
Annicchiarico, 8, 82, 194, 201
anova_ind, 9, 12, 66, 71
anova_joint, 11, 66, 71
arrange_ggplot, 13
as.lpcor, 14
as.split_factors (split_factors), 198
av_dev (utils_stats), 217

barplots, 15
bind_cv, 19, 121

can_corr, 20
ci_mean (utils_stats), 217
clustering, 22
colindiaq, 24
colnames_to_lower (utils_rows_cols), 213
colnames_to_title (utils_rows_cols), 213
colnames_to_upper (utils_rows_cols), 213
colors, 201
column_exists (utils_rows_cols), 213
column_to_first (utils_rows_cols), 213
column_to_last (utils_rows_cols), 213
comb_vars, 26

cor, 108, 112
corr_ci, 27
corr_coef, 28
corr_plot, 29
corr_ss, 32
corr_stab_ind, 34
covcor_design, 35
cv (utils_stats), 217
cv_ammi, 36, 40, 43, 121
cv_ammi, 38, 39, 43, 121, 159, 160
cv_blup, 38, 40, 41, 121
cv_by (utils_stats), 217
data_alpha, 43
data_g, 44
data_ge, 45
data_ge2, 46
desc_stat, 47
freq_table (utils_stats), 217

ecovalence, 9, 51, 66, 71, 77, 81, 82, 194, 201
equivdissimilarity, 52
extract_number (utils_num_str), 208
extract_string (utils_num_str), 208

cor, 108, 112
corr_ci, 27
corr_coef, 28
corr_plot, 29
corr_ss, 32
corr_stab_ind, 34
covcor_design, 35
cv (utils_stats), 217
cv_ammi, 36, 40, 43, 121
cv_ammi, 38, 39, 43, 121, 159, 160
cv_blup, 38, 40, 41, 121
cv_by (utils_stats), 217
data_alpha, 43
data_g, 44
data_ge, 45
data_ge2, 46
desc_stat, 47
freq_table (utils_stats), 217

ecovalence, 9, 51, 66, 71, 77, 81, 82, 194, 201
equivdissimilarity, 52
extract_number (utils_num_str), 208
extract_string (utils_num_str), 208

cor, 108, 112
corr_ci, 27
corr_coef, 28
corr_plot, 29
corr_ss, 32
corr_stab_ind, 34
covcor_design, 35
cv (utils_stats), 217
cv_ammi, 36, 40, 43, 121
cv_ammi, 38, 39, 43, 121, 159, 160
cv_blup, 38, 40, 41, 121
cv_by (utils_stats), 217
data_alpha, 43
data_g, 44
data_ge, 45
data_ge2, 46
desc_stat, 47
freq_table (utils_stats), 217

ecovalence, 9, 51, 66, 71, 77, 81, 82, 194, 201
equivdissimilarity, 52
extract_number (utils_num_str), 208
extract_string (utils_num_str), 208

234
predict.gge, 157
predict.performs_ammi, 159
predict.waas, 160
predict.waasb, 161
print.AMMI_indexes, 162
print.Annicchiario, 163
print.anova_ind, 164
print.anova_joint, 164
print.can_cor, 165
print.colindiag, 165
print.corr_coef, 166
print.ecovalence, 166
print.env_dissimilarity, 167
print.Fox, 169
print.ge_factanal, 170
print.ge_reg, 171
print.Huehn, 172
print.lpcor, 172
print.mgidi, 173
print.mtsi, 173
print.path_coeff, 174
print.performs_ammi, 175
print.Schmildt, 176
print.Shukla, 177
print.superiority, 178
print.Thennarasu, 179
print.waas, 180
print.waas_means, 181
print.waasb, 182
random_na (utils_na), 207
range_data (utils_stats), 217
rbind_fill, 185
remove_class (utils_class), 205
remove_cols (utils_rows_cols), 213
remove_cols_na (utils_na), 207
remove_rows (utils_rows_cols), 213
remove_rows_na (utils_na), 207
remove_space (utils_num_str), 208
remove_strings (utils_num_str), 208
reorder_cols (utils_rows_cols), 213
reorder_cormat, 185
replace_na (utils_na), 207
replace_number (utils_num_str), 208
replace_string (utils_num_str), 208
resca, 186, 232
Resende_indexes, 66, 71, 82, 188
residual_plots, 115, 125, 139, 141, 189
resp_surf, 191
round_cols (utils_num_str), 208
Schmildt, 193
sd_amo (utils_stats), 217
sd_by (utils_stats), 217
sd_pop (utils_stats), 217
select_cols, 195
select_cols (utils_rows_cols), 213
select_cols_na (utils_na), 207
select_first_col (utils_rows_cols), 213
Select_helper, 194
select_last_col (utils_rows_cols), 213
select_non_numeric_cols (utils_rows_cols), 213
select_numeric_cols (utils_rows_cols), 213
select_rows (utils_rows_cols), 213
select_rows_na (utils_na), 207
sem (utils_stats), 217
sem_by (utils_stats), 217
set_class (utils_class), 205
Shukla, 66, 71, 82, 196
skew (utils_stats), 217
solve_svd, 197
split_factors, 198
stars_pval, 199
tibble::print(), 162–166, 168–178, 180–184
tidy_strings (utils_num_str), 208
tidy_sym (utils_mat), 206
title_case_only (Select_helper), 194
to_factor, 203
transparent_color (themes), 201
tukey_hsd, 204
TukeyHSD, 204
union_var (Select_helper), 194
upper_case_only (Select_helper), 194
utils_class, 205
utils_mat, 206
utils_na, 207
utils_num_str, 208
utils_rows_cols, 213
utils_stats, 217
valid_n (utils_stats), 217
var_amo (utils_stats), 217
var_pop (utils_stats), 217

waas.means, 114, 152, 222, 229
waasb, 61, 66, 71, 106, 114, 152, 161, 222, 224, 230, 232
width_greater_than (Select_helper), 194
width_less_than (Select_helper), 194
width_of (Select_helper), 194
wsmp, 113, 231