Package 'RAP'

March 30, 2018

Type Package

Title R Analytical Pipeline (RAP)

Version 0.0.0.9000

Date 2018-04-01

Description The R Analytical Pipeline (RAP), commissioned by the Generation Challenge Program (GCP), is being developed to streamline the analysis of typical plant breeding experiments, including both phenotypic analysis of field trials and genotype by environment analysis. Some functions have been created to be used in conjunction with the R package 'asreml' for the 'ASReml' software, which can be obtained upon purchase from 'VSN' international (http://www.vsni.co.uk/software/asreml).

License GPL-3

Encoding UTF-8

LazyData true

RoxygenNote 6.0.1

Depends R (>= 3.4.4)

Imports desplot(>= 1.3),
dplyr(>= 0.7.4),
emmeans(>= 1.1.2),
fields(>= 9.6),
ggplot2(>= 2.2.1),
knitr(>= 1.20),
lattice(>= 0.10-35),
lme4(>= 1.1-15),
maps(>= 3.2.0),
methods(>= 3.4.4),
qtl(>= 1.42-8),
reshape2(>= 1.4.3),
rlang(>= 0.2.0),
SpATS(>= 1.0-7),
tibble(>= 1.4.2),
xtable(>= 1.8.2)

Suggests asreml(== 3.0),
rmarkdown(>= 1.9),
testthat(>= 2.0.0)

VignetteBuilder knitr

R topics documented:

AMMI *S3 class AMMI*

Description

Function for creating objects of S3 class AMMI.
print, summary, plot and report methods are available.

Usage

```
createAMMI(envScores, genoScores, importance, anova, fitted, trait, envMean,
  genoMean, overallMean)
```

Arguments

envScores	A matrix containing environmental scores.
genoScores	A matrix containing genotypic scores.
importance	A data.frame containing the importance of the principal components.
anova	A data.frame containing anova scores of the AMMI analysis.
fitted	A matrix containing fitted values from the AMMI model.
trait	A character string indicating the analyzed trait.
envMean	A numerical vector containing the environmental means.
genoMean	A numerical vector containing the genotypic means.
overallMean	A numerical value containing the overall mean.

Author(s)

Bart-Jan van Rossum

See Also

plot.AMMI, report.AMMI

FW *S3 class FW*

Description

Function for creating objects of S3 class FW (Finlay-Wilkinson).
print, summary, plot and report methods are available.

Usage

```
createFW(estimates, anova, envEffs, trait, nGeno, nEnv, TD, fittedGeno, tol,
  iter)
```

Arguments

estimates	A data.frame containing the estimated values.
anova	A data.frame containing anova scores of the FW analysis.
envEffs	A data.frame containing the environmental effects.
trait	A character value indicating the analysed trait.
nGeno	A numerical value containing the number of genotypes in the analysis.
nEnv	A numerical value containing the number of environments in the analysis.
TD	The object of class TD on which the analysis was performed.
fittedGeno	The fitted values for the genotypes.
tol	A numerical value containing the tolerance used during the analysis.
iter	A numerical value containing the number of iterations for the analysis to converge.

Author(s)

Bart-Jan van Rossum

See Also

plot.FW, report.FW

---

getMeta *Extract metadata from TD objects*

Description

Function for extracting metadata as a data.frame from objects of class TD. Location, data, design, latitude, longitude, plotWidth and plotLength for all trials in TD will be extracted and return in the form of a data.frame.

Usage

```
getMeta(TD)
```

Arguments

TD	An object of class TD.

Value

A data.frame containing the metadata for all trials in TD.

gxeAmmi *AMMI analysis*

Description

This function fits a model which involves the Additive Main effects (i.e. genotype and trial) along with the Multiplicative Interaction effects of principal component analysis (PCA).

Usage

```
gxeAmmi(TD, trials = names(TD), trait, nPC = 2, center = TRUE,
  scale = FALSE)
```

Arguments

TD	An object of class TD.
trials	A character string specifying the trials to be analyzed. If not supplied all trials are used in the analysis.
trait	A character string specifying the trait to be analyzed.
nPC	An integer specifying the number of principal components used as multiplicative term of genotype-by-trial interaction.
center	Should the variables be shifted to be zero centered?
scale	Should the variables be scaled to have unit variance?

Value

An object of class AMMI, a list containing:

envScores	A matrix with environmental scores.
genoScores	A matrix with genotypic scores.
importance	A data.frame containing the importance of the principal components.
anova	A data.frame containing anova scores of the AMMI analysis.
fitted	A matrix containing fitted values from the AMMI model.
trait	A character string containing the analyzed trait.
envMean	A numerical vector containing the environmental means.
genoMean	A numerical vector containing the genotypic means.
overallMean	A numerical value containing the overall mean.

References

Gauch H.G. (1992) Statistical Analysis of Regional Yield Trials: AMMI Analysis of Factorial Designs. Elsevier, Amsterdam.

See Also

AMMI, plot.AMMI, report.AMMI

Examples

```
## Run AMMI analysis on TDMaize.
geAmmi <- gxeAmmi(TD = TDMaize, trait = "yld")
## Summarize results.
summary(geAmmi)
## Create a biplot of genotypes and environment interaction with PC1 and PC2.
plot(geAmmi, plotType = "AMMI2")
## Create a pdf report summarizing the results.
## Not run:
report(geAmmi, outfile = "./testReports/reportAmmi.pdf")

## End(Not run)
```

---

gxeFw *Finlay-Wilkinson analysis*

Description

This function performs a Finlay-Wilkinson analysis of data classified by two factors.

Usage

```
gxeFw(TD, trials = names(TD), trait, maxIter = 15, tol = 0.001,
  sorted = c("ascending", "descending", "none"))
```

Arguments

TD	An object of class TD.
trials	A character string specifying the trials to be analyzed. If not supplied all trials are used in the analysis.
trait	A character string specifying the trait to be analyzed.
maxIter	An integer specifying the maximum number of iterations in the algorithm.
tol	A positive numerical value specifying convergence tolerance of the algorithm.
sorted	A character string specifying the sorting order of the estimated values in the output.

Value

An object of class FW, a list containing:

estimates	A data.frame containing the estimated values.
anova	A data.frame containing anova scores of the FW analysis.
envEffs	A data.frame containing the environmental effects.
TD	The object of class TD on which the analysis was performed.
fittedGeno	A numerical vector containing the fitted values for the genotypes.
trait	A character string containing the analyzed trait.
nGeno	A numerical value containing the number of genotypes in the analysis.

nEnv	A numerical value containing the number of environments in the analysis.
tol	A numerical value containing the tolerance used during the analysis.
iter	A numerical value containing the number of iterations for the analysis to converge.

References

Finlay, K.W. & Wilkinson, G.N. (1963). The analysis of adaptation in a plant-breeding programme. Australian Journal of Agricultural Research, 14, 742-754.

See Also

FW, plot.FW, report.FW

Examples

```
## Run Finlay-Wilkinson analysis on TDMaize.
geFW <- gxeFw(TDMaize, trait = "yld")
## Summarize results.
summary(geFW)
## Create a scatterplot of the results.
plot(geFW, plotType = "scatter")
## Not run:
## Create a report summarizing the results.
report(geFW, outfile = "./testReports/reportFW.pdf")

## End(Not run)
```

---

gxeMegaEnv *Form mega-environments based on winning genotypes from an AMMI model*

Description

This function fits an AMMI model and then using the fitted values produces a new factor based on the winning genotype in each environment. This factor is added as a column megaEnv to the input data. If a column megaEnv already exists the existing data is overwritten with a warning.

Usage

```
gxeMegaEnv(TD, trials = names(TD), trait, method = c("max", "min"),
  sumTab = TRUE)
```

Arguments

TD	An object of class TD.
trials	A character string specifying the trials to be analyzed. If not supplied all trials are used in the analysis.
trait	A character string specifying the trait to be analyzed.
method	A character string indicating the criterion to determine the best genotype per environment, either "max" or "min".
sumTab	Should a summary table be added as an attribute to the output and be printed?

Value

The input object of class TD with an added extra column megaEnv.

Examples

```
## Calculate mega-environments for TDMaize and print a summary of the results.
TDmegaEnv <- gxeMegaEnv(TD = TDMaize, trait = "yld")
## Calculate new mega-environments based on the genotypes with the lowest
## value per environment.
TDmegaEnv2 <- gxeMegaEnv(TD = TDmegaEnv, trait = "yld", method = "min")
```

---

gxeStability *Calculate stability coefficients for genotype-by-environment data*

Description

This function calculates different measures of stability, the cultivar-superiority measure of Lin & Binns (1988), Shukla's (1972) stability variance and Wricke's (1962) ecovalence.

Usage

```
gxeStability(TD, trials = names(TD), trait, method = c("superiority",
  "static", "wricke"), bestMethod = c("max", "min"),
  sorted = c("descending", "ascending", "none"))
```

Arguments

TD	An object of class TD.
trials	A character string specifying the trials to be analyzed. If not supplied all trials are used in the analysis.
trait	A character string specifying the trait to be analyzed.
method	A character vector specifying the measures of stability to be calculated. Options are superiority (cultivar-superiority measure), static (Shukla's stability variance) or wricke (wricke's ecovalence).
bestMethod	A character string specifying the criterion to define the best genotype. Either "max" or "min".
sorted	A character string specifying the sorting order of the results.

Value

An object of class stability, a list containing:

superiority	A data.frame containing values for the cultivar-superiority measure of Lin and Binns.
static	A data.frame containing values for Shukla's stability variance.
wricke	A data.frame containing values for Wricke's ecovalence.
trait	A character string indicating the trait that has been analyzed.

References

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

Shukla, G.K. 1972. Some statistical aspects of partitioning genotype-environmental components of variability. Heredity 29:237-245

Wricke, G. Uber eine method zur erfassung der okologischen streubreit in feldversuchen. Zeitschrift für Pflanzenzucht, v. 47, p. 92-96, 1962

See Also

stability, plot.stability, report.stability

Examples

```
## Compute three stability measures for TDMaize.
geStab <- gxeStability(TD = TDMaize, trait = "yld")
## Summarize results.
summary(geStab)
## Create plot of the computed stability measures against the means.
plot(geStab)
## Not run:
## Create a .pdf report summarizing the stability measures.
report(geStab, outfile = "./testReports/reportStability.pdf")

## End(Not run)

## Compute Wricke's ecovalance for TDMaize with minimal values for yield as
## the best values. Sort results in ascending order.
geStab2 <- gxeStability(TD = TDMaize, trait = "yld", method = "wricke",
                        bestMethod = "min", sorted = "ascending")
summary(geStab2)
```

---

gxeTable *Compute BLUPS and based on a set of mega-environments*

Description

This function calculates predicted means (BLUPS) and associated standard errors based on a set of mega-environments.

Usage

```
gxeTable(TD, trials = names(TD), trait, useYear = FALSE,
  engine = c("lme4", "asreml"), ...)
```

Arguments

TD	An object of class TD.
trials	A character string specifying the trials to be analyzed. If not supplied all trials are used in the analysis.

trait	A character string specifying the trait to be analyzed.
useYear	Should year be used for modelling (as years within trials). If TRUE TD should contain a column "year".
engine	A character string specifying the engine used for modeling. Either "lme4" or "asreml".
...	Further parameters passed to either asreml or lmer.

Value

A list consisting of two data.frames, predictedValue containing BLUPs per genotype per mega-environment and standardError containing standard errors for those BLUPs.

Examples

```
## Compute mega-environments for TDMaize.
TDMegaEnv <- gxeMegaEnv(TD = TDMaize, trait = "yld", sumTab = FALSE)
## Compute BLUPS and standard errors for those mega-environments.
geTab <- gxeTable(TD = TDMegaEnv, trait = "yld")
```

---

gxeVarComp *Selects the best variance-covariance model for a set of trials*

Description

This function selects the best covariance structure for genetic correlations between trials. It fits a range of variance-covariance models to compare (identity, compound symmetry, diagonal, heterogeneous compound symmetry, first order factor analysis, second order factor analysis, unstructured), and selects the best one using a goodness-of-fit criterion.

Usage

```
gxeVarComp(TD, trials = names(TD), trait, engine = c("lme4", "asreml"),
  criterion = c("BIC", "AIC"), ...)
```

Arguments

TD	An object of class TD.
trials	A character string specifying the trials to be analyzed. If not supplied all trials are used in the analysis.
trait	A character string specifying the trait to be analyzed.
engine	A character string specifying the engine used for modeling. Either "lme4" or "asreml".
criterion	A string specifying a goodness-of-fit criterion. Either "AIC" or "BIC".
...	Further arguments to be passed to asreml.

Value

An object of class varComp, a list object containing:

SSA	An object of class SSA containing the best fitted model.
choice	A character string indicating the best fitted model.
summary	A data.frame with a summary of the fitted models.
vcov	The covariance matrix of the best fitted model.
criterion	A character string indicating the goodness-of-fit criterion used for determinening the best model, either "AIC" or "BIC".
engine	A character string containing the engine used for the analysis.

Note

If `engine = "lme4"`, only the compound symmetry model can be fitted.

Examples

```
## Select the best variance-covariance model using lme4 for modeling.
geVarComp <- gxeVarComp(TD = TDMaize, trait = "yld")
## Summarize results.
summary(geVarComp)
## Not run:
## Create a pdf report summarizing the results.
report(geVarComp, outfile = "./testReports/reportVarComp.pdf")

## End(Not run)

## Not run:
## Select the best variance-covariance model using asreml for modeling.
## Use BIC as a goodness-of-fit criterion.
geVarComp2 <- gxeVarComp(TD = TDMaize, trait = "yld", engine = "asreml",
                         criterion = "BIC")
summary(geVarComp2)
## Create a heatmap of the correlation matrix for the best model.
plot(geVarComp2)

## End(Not run)
```

---

multiQTL *S3 class multiQTL*

Description

Function for creating objects of S3 class multiQTL.
summary, plot and report methods are available.

Usage

```
createMultiQTL(qtl, QTLDet, selection, thr)
```

Arguments

qtl	A fitted multi QTL model.
QTLDet	The object of class QTLDet used as base for fitting the QTL model.
selection	A character string indictating the type of selection used for selecting the markers in the final model.
thr	A numerical value indicating the threshold for dropping terms in the backwards elemination process.

Author(s)

Bart-Jan van Rossum

See Also

plot.multiQTL, report.multiQTL

---

multiQTLFit *Fit an additive multi QTL model*

Description

An additive multi QTL model is fitted based on the peaks in the QTLDet object. Fitting is done using the fitqtl function in the qtl package. After fitting the model backward elemination is done until all markers in the model have a significant P-value.

Usage

```
multiQTLFit(QTLDet, selection = c("backward", "none"), thr = 0.05, ...)
```

Arguments

QTLDet	An object of class QTLDet.
selection	A character string indicating whether backward selection should be applied or no selection at all.
thr	A positive numerical value indicating the threshold for dropping terms in the backwards elemination process.
...	Further parameters to be passed on to underlying functions used for qtl detection, scanone when type is "MR" or "SIM" and cim when type is "CIM".

Value

An object of class multiQTL, a list containing:

scores	A data.frame containing the lod scores.
peaks	A data.frame containing the peaks found.
type	A character string indicating the type of QTLDetection performed.
cross	An object of class cross in the qtl package on which the analysis has been performed.
trait	A character string indicating the trait for which the analysis is done.
info	A list containing information on the settings used for QTL Detection, i.e. step, threshold and window.

References

Broman et al. (2003) R/qtl: QTL mapping in experimental crosses. Bioinformatics 19:889-890

See Also

fitqtl

Examples

```
## Read the data
F2 <- qtl::read.cross(format="csv",
                      file = system.file("extdata",
                                         "F2_maize_practical3_ex2.csv",
                                         package = "RAP"),
                      genotypes = c("AA", "AB", "BB"),
                      alleles = c("A", "B"), estimate.map = FALSE)
## Perform QTL detection using simple interval mapping.
QTLDet <- QTLDetect(cross = F2, trait = "trait", type = "SIM")
## Fit a multi QTL model.
multiFit <- multiQTLFit(QTLDet)
## Summarize results.
summary(multiFit)
## Not run:
## Create a pdf report summarizing results.
report(multiFit, outfile = "./testReports/reportMultiQTLFit.pdf")

## End(Not run)
```

---

multMissing *Multmissing procedure*

Description

This function estimates missing values for units in a multivariate data set, using an iterative regression technique.

Usage

```
multMissing(Y, maxIter = 10, naStrings = NULL)
```

Arguments

Y	A matrix, data.frame or vector of multivariate data.
maxIter	An integer specifying the maximum number of iterations.
naStrings	A character vector of strings which are to be interpreted as NA values.

Details

Initial estimates of the missing values in each variate are formed from the variate means using the values for units that have no missing values for any variate. Estimates of the missing values for each variate are then recalculated as the fitted values from the multiple regression of that variate on all the other variates. When all the missing values have been estimated the variate means are recalculated. If any of the means differs from the previous mean by more than a tolerance (the initial standard error divided by 1000) the process is repeated, subject to a maximum number of repetitions defined by `maxIter` option. The default maximum number of iterations (10) is usually sufficient when there are few missing values, say two or three. If there are many more, 20 or so, it may be necessary to increase the maximum number of iterations to around 30. The method is similar to that of Orchard & Woodbury (1972), but does not adjust for bias in the variance-covariance matrix as suggested by Beale & Little (1975).

Value

An object of the same class as the input `Y` with the missing values replaced by their estimates.

References

Beale, E.M.L. & Little, R.J.A. (1975). Missing values in multivariate analysis. Journal of the Royal Statistical Society, Series B, 37, 129-145.

Orchard, T. & Woodbury, M.A. (1972). A missing information principle: theory and applications. In: Proceedings of the 6th Berkeley Symposium in Mathematical Statistics and Probability, Vol I, 697-715.

Examples

```
M <- matrix(c("1", "2", "3", NA, "b", "5", "6",
              "6", "5", "b", NA, "3", "2", "1"), nrow = 7, ncol = 2)
## Estimate missing values treating "b" as NA.
multMissing(M, naStrings = "b")
```

`outlierSSA` *Identifying outliers in objects of class SSA*

Description

Function to identify observations with standardized residuals exceeding `rLimit`. If not provided `rLimit` is computed as `qnorm(1 - 0.5 / rDf)` where `rDf` are the residual degrees of freedom for the model. This value is then restricted to the interval 2..4. Alternatively a custom limit may be provided.
A summary is printed of outliers and observations that have the same value for `commonFactors`. The latter ones will be marked as similar to distinguish them from the former ones.

Usage

```
outlierSSA(SSA, trial = NULL, traits, rLimit = NULL, commonFactors = NULL)
```

Arguments

SSA	An object of class SSA.
trial	A character string specifying the trial for which outliers should be identified. If NULL and SSA contains only one trial that trial is used.
traits	A character vector specifying the names of the traits for which outliers should be identified.
rLimit	A numerical value used for determining when a value is considered an outlier. All observations with standardized residuals exceeding rLimit will be marked as outliers.
commonFactors	A character vector specifying the names of columns in TD used for selecting observations that are similar to the outliers. If commonFactors = NULL only outliers are reported and no similar observations.

Value

A data.frame containing logical values indicating if the observation is an outlier.

Examples

```
## Fit a model using lme4.
myModel <- STRunModel(TD = TDHeat05, traits = "yield" ,
                      design = "res.rowcol", engine = "lme4")
## Detect outliers in the standardized residuals of the fitted model.
outliers <- outlierSSA(SSA = myModel, traits = "yield")
```

---

plot.AMMI *Plot function for class AMMI*

Description

Two types of biplot can be made. A biplot of genotype and environment means vs PC1 (AMMI1) or a biplot of genotypes and environment interaction with PC1 and PC2 (AMMI2).

Usage

```
## S3 method for class 'AMMI'
plot(x, ..., plotType = c("AMMI1", "AMMI2"), scale = 1,
  col = c("orange3", "navyblue"))
```

Arguments

x	An object of class AMMI
...	Further graphical parameters passed on to actual plot function.
plotType	A character string indicating which plot should be made. Either "AMMI1" or "AMMI2" for an AMMI1 biplot (genotype and environment means vs PC1) or an AMMI2 biplot (genotypes and environment interaction with PC1 and PC2) respectively.

scale	A numerical value. The variables are scaled by lambda ^ scale and the observations by lambda ^ (1 - scale) where lambda are the singular values computed by princomp in gxeAmmi. Normally 0 <= scale <= 1, and a warning will be issued if the specified scale is outside this range.
col	A character vector with plot colors for genotype and environment.

Value

A biplot depending on plotType.

Examples

```
## Run AMMI analysis.
geAmmi <- gxeAmmi(TD = TDMaize, trait = "yld")
## Create a biplot of genotype and environment means vs PC1.
plot(geAmmi)
## Create a biplot of genotypes and environment interaction with PC1 and PC2.
plot(geAmmi, plotType = "AMMI2")
```

---

plot.FW *Plot function for class FW*

Description

Three types of plot can be made. A scatter plot for genotypic mean, mse and sensitivity, a line plot with fitted lines for each genotype and a trellis plot with individual slopes per genotype (for max 64 genotypes). If there are more than 64 genotypes only the first 64 are plotted in the trellis plot.

Usage

```
## S3 method for class 'FW'
plot(x, ..., plotType = c("scatter", "line", "trellis"),
  sorted = c("ascending", "descending", "none"))
```

Arguments

x	An object of class FW.
...	Further graphical parameters passed on to actual plot function.
plotType	A character string indicating which plot should be made. Either "scatter", "line" or "trellis" for creating a scatter plot of genotypic means, mse and sensitivities, a plot of fitted lines for each genotype or a trellis plot of the individual genotype slopes respectively.
sorted	A character string specifying whether the results should be sorted in an increasing (or decreasing) order of sensitivities.

Value

A plot depending on plotType.

Examples

```
## Run Finlay-Wilkinson analysis.
geFW <- gxeFw(TD = TDMaize, trait = "yld")
## Create a scatter plot.
plot(geFW)
## Create a line plot.
plot(geFW, plotType = "line")
## Create a line plot.
plot(geFW, plotType = "trellis")
```

`plot.multiQTL` *Plot function for class multiQTL*

Description

Plotting function for objects of class multiQTL. Plots the estimates of the QTLs in the final model with their respective confidence intervals.

Usage

```
## S3 method for class 'multiQTL'
plot(x, ...,
  main = "QTL estimates and confidence intervals")
```

Arguments

`x`	An object of class multiQTL
`...`	Further graphical parameters. Currently not used.
`main`	A character string used as overall title for the plot.

Examples

```
## Read the data
F2 <- qtl::read.cross(format="csv",
                      file = system.file("extdata",
                                         "F2_maize_practical3_ex2.csv",
                                         package = "RAP"),
                      genotypes = c("AA", "AB", "BB"),
                      alleles = c("A", "B"), estimate.map = FALSE)
## Perform QTL detection using simple interval mapping.
QTLDet <- QTLDetect(cross = F2, trait = "trait", type = "SIM")
## Fit a multi QTL model.
multiFit <- multiQTLFit(QTLDet)
## Plot the results.
plot(multiFit)
```

plot.QTLDet *Plot function for class QTLDet*

Description

Function for creating a manhattan plot for objects of class QTLDet.

Usage

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

Arguments

x	An object of class QTLDet
...	Not used

plot.SSA *Plot function for class SSA*

Description

This function draws either four base plots:

- A histogram of the residuals
- A normal Q-Q plot
- A residuals vs fitted values plot
- An absolute residuals vs fitted values plot

or five or (in case SpATS is used for modelling) six spatial plots:

- A spatial plot of the raw data
- A spatial plot of the fitted data
- A spatial plot of the residuals
- A spatial plot of the estimated spatial trend (SpATS only)
- A spatial plot of the BLUEs or BLUPs
- A histogram of the BLUEs or BLUPs

Spatial plots can only be made if the data contains both row and column information.

Usage

```
## S3 method for class 'SSA'
plot(x, ..., trial = NULL, trait = NULL, what = NULL,
  plotType = c("base", "spatial"))
```

Arguments

x	An object of class SSA.
...	Further graphical parameters (see xyplot for details).
trial	A character string indicating the trial to plot. If trial = NULL and only one trial is modelled this trial is plotted.
trait	A character string indicating the trait to plot. If trait = NULL and only one trait is modelled this trait is plotted.
what	A character string indicating whether the fitted model with genotype as fixed or genotype as random factor should be plotted. If x contains only one model this model is chosen automatically.
plotType	A Character string indicating whether base plots or spatial plots should be made.

See Also

SSA

Examples

```
## Run a single trait analysis using SpATS.
myModel <- STRunModel(TD = TDHeat05, design = "res.rowcol", traits = "yield")
## Create base plots.
plot(myModel, what = "fixed", plotType = "base")
## Create spatial plots.
plot(myModel, what = "fixed", plotType = "spatial")
```

---

plot.stability *Plot function for class stability*

Description

Function for creating scatter plots of computed stability measures against the means.

Usage

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

Arguments

x	An object of class stability.
...	Further arguments to be passed on to underlying plot functions.

Value

Plots of stability measures against means.

Examples

```
## Compute three stability measures for TDMaize.
geStab <- gxeStability(TD = TDMaize, trait = "yld")
## Create scatter plots of the computed stability measures against the means.
plot(geStab)
```

`plot.TD` *Plot function for class TD*

Description

Plotting function for objects of class TD. Plots either the layout of the different trials within the TD object or locates the trials on a map. Mapping the trials is done based on lattitude and longitude that can be added when creating an object of class TD. The countries in which the trials are located will be plotted on a single map and the location of the trials will be indicated on this map.

Usage

```
## S3 method for class 'TD'
plot(x, ..., trials = names(x), plotType = c("layout", "map"))
```

Arguments

`x`	An object of class TD.
`...`	Further graphical parameters. Currently not used.
`trials`	A character vector indicating the trials to be plotted.
`plotType`	A character string indicating which plot should be made. This can be "layout" for a plot of the field layout for the diffent trials in the TD object. This is only possible if the data contains row and column information. Alternatively, for `plotType = "map"` a plot will be made depicting the trials on a country map. This is only possible if lattitude and longitude of the trials are available.

`plot.varComp` *Plot function for class varComp*

Description

Function for plotting a heatmap of the correlation matrix for objects of class varComp.

Usage

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

Arguments

`x`	An object of class varComp
`...`	Not used

Examples

```
## Not run:
## Select the best variance-covariance model using asreml for modeling.
geVarComp <- gxeVarComp(TD = TDMaize, trait = "yld", engine = "asreml")
## Create a heatmap of the correlation matrix for the best model.
plot(geVarComp)

## End(Not run)
```

QTLDet *S3 class QTLDet*

Description

Function for creating objects of S3 class QTLDet.
print, summary, plot and report methods are available.

Usage

```
createQTLDet(scores, peaks, type, cross, trait, info)
```

Arguments

scores	A data.frame containing the lod scores.
peaks	A data.frame containing the peaks found.
type	A character string indicating the type of QTL detection performed.
cross	An object of class cross in the qtl package.
trait	A character string indicating the trait for which the analysis is done.
info	A list containing information on the settings used for QTL detection, i.e. step, threshold and window.

Author(s)

Bart-Jan van Rossum

See Also

plot.QTLDet, report.QTLDet

QTLDetect *QTL detection*

Description

This function is essentially a wrapper for scanone and cim in the qtl package. Depending on type one of these functions is used for QTL detection. After this is done, from the set of candidate QTLs that are returned proper peaks are selected by an iterative process using the threshold and window provided. All resulting peaks will have a LOD score above threshold and the distance between pairs of peaks will always be at least the value given as window.

Usage

```
QTLDetect(cross, trait, type = c("MR", "SIM", "CIM"), step = 5, thr = 3,
  window = 30, ...)
```

Arguments

cross	An object of class cross created by the qtl package.
trait	A character string indicating the trait to be analyzed.
type	A character string indicating the type of QTL detection to be performed. Either "MR" (Marker Response), "SIM" (Simple Interval Mapping) or "CIM" (Composite Interval Mapping)
step	A numerical value indicating the maximum distance (in cM) between positions at which the genotype probabilities are calculated. If step = 0 probabilities are only calculated at the marker locations.
thr	A numerical value indicating a lower threshold for the lodscore of the peaks.
window	A numerical value indicating the window (in cM) used when selecting peaks.
...	Further parameters to be passed on to underlying functions used for qtl detection, scanone when type is "MR" or "SIM" and cim when type is "CIM".

Value

An object of class QTLDet, a list containing:

scores	A data.frame containing the lod scores.
peaks	A data.frame containing the peaks found.
type	A character string indicating the type of QTL detection performed.
cross	An object of class cross in the qtl package.
trait	A character string indicating the trait for which the analysis is done.
info	A list containing information on the settings used for QTL detection, i.e. step, threshold and window.

References

Broman et al. (2003) R/qtl: QTL mapping in experimental crosses. Bioinformatics 19:889-890

See Also

scanone, cim

Examples

```
## Read the data.
F2 <- qtl::read.cross(format="csv",
                      file = system.file("extdata",
                                         "F2_maize_practical3_ex2.csv",
                                         package = "RAP"),
                      genotypes = c("AA", "AB", "BB"),
                      alleles = c("A", "B"), estimate.map = FALSE)
## Perform a composite interval mapping for detecting QTLs.
QTLDet <- QTLDetect(cross = F2, trait = "trait", type = "CIM")
## Summarize results.
summary(QTLDet)
## Create a manhattan plot of the results.
plot(QTLDet)
## Not run:
## Create a pdf report summarizing the results.
report(QTLDet, outfile = "./testReports/reportQTLDectection.pdf")

## End(Not run)

## Perform a simple interval mapping for detecting QTLs.
## Choose custom step, threshold and window sizes.
QTLDet2 <- QTLDetect(cross = F2, trait = "trait", type = "SIM", step = 15,
                     thr = 2.5, window = 50)
summary(QTLDet2)
```

QTLMapQC *Quality Control and Cleaning of a cross object*

Description

Function for performing quality control and cleaning of a cross object. Quality control is done in several subsequent steps:

1. Markers with a fraction of missing values higher than `missMrk` are removed.
2. Duplicate markers are removed.
3. Individuals with a fraction of missing values higher than `missInd` are removed.
4. Markers that show evidence of segregation distortion (P-value below `segDistortion`) are removed.
5. Markers which might have been switched (with threshold `recombination` are removed. See also `checkAlleles`).
6. The map is reestimated based on the observed markers.
7. Individuals with a fraction of crossovers higher than `crossover` are removed.

Steps 1, 3, 4, 5 and 7 are only performed if their respective threshold values are positive. Setting them to 0 suppresses the corresponding check.
Steps 2 and 6 are performed if respectively `removeDuplicates` and `reestimateMap` are `TRUE`.

Usage

```
QTLMapQC(cross, missMrk = 0.05, missInd = 0.05, removeDuplicates = TRUE,
  segDistortion = 0.001, recombination = 3, reestimateMap = FALSE,
  crossover = 0.2)
```

Arguments

cross	An object of class cross created by the qtl package.
missMrk	A numerical value between 0 and 1 indicating the maximum allowed fraction of missing values per marker. Markers with a fraction of missing values above missMrk will be removed.
missInd	A numerical value between 0 and 1 indicating the maximum allowed fraction of missing values per individual. Individuals with a fraction of missing values above missMrk will be removed.
removeDuplicates	Should duplicate markers be removed?
segDistortion	A numerical value between 0 and 1 used a threshold for Mendelian segregation. Markers with a P-value below segDistortion will be removed.
recombination	A positive numerical value used a threshold for checking recombination between pairs of markers.
reestimateMap	Should the map be reestimated based on the observed markers?
crossover	A numerical value between 0 and 1 indicating the maximum allowed fraction of crossovers per individual. Individuals with a fraction of crossovers above crossover will be removed.

Value

A cleaned version of the input cross object after markers and individuals have been removed that are outside the respective thresholds.

References

Broman et al. (2003) R/qtl: QTL mapping in experimental crosses. Bioinformatics 19:889-890

See Also

nmissing, drop.markers, drop.dupmarkers, geno.table, nmissing, checkAlleles, replace.map, countXO

Examples

```
## Read the data.
F2 <- qtl::read.cross(format="csv",
                      file = system.file("extdata",
                                         "F2_maize_practical3_ex2.csv",
                                         package = "RAP"),
                      genotypes = c("AA", "AB", "BB"),
                      alleles = c("A", "B"), estimate.map = FALSE)
## Run quality control.
F2QC <- QTLMapQC(F2)
## Compare cross object before and after cleaning.
summary(F2)
```

```
summary(F2QC)

## Run quality control: only remove markers with a fraction of missing
## values higher than 0.02
F2QC2 <- QTLMapQC(F2, missMrk = 0.02, missInd = 0, removeDuplicates = FALSE,
                  segDistortion = 0, recombination = 0, crossover = 0)
summary(F2QC2)
```

RAP *RAP package*

Description

RAP

---

report *Base method for creating a report*

Description

Base method for creating a .pdf and .tex report from an R object

Usage

```
report(x, ...)
```

Arguments

x	An R object
...	Further arguments to be passed on to specific report functions.

---

report.AMMI *Report method for class AMMI*

Description

A pdf report will be created containing a summary of an AMMI model. Simultaneously the same report will be created as a tex file.

Usage

```
## S3 method for class 'AMMI'
report(x, ..., outfile = NULL)
```

Arguments

x	An object of class AMMI.
...	Further arguments passed on from other functions - not used yet.
outfile	A character string, the name and location of the output .pdf and .tex file for the report. If NULL a report with a default name will be created in the current working directory.

Value

A pdf and tex report.

Examples

```
## Run AMMI analysis on TDMaize.
geAmmi <- gxeAmmi(TD = TDMaize, trait = "yld")
## Not run:
## Create a pdf report summarizing the results.
report(geAmmi, outfile = "./testReports/reportAmmi.pdf")

## End(Not run)
```

---

report.cross *Report method for class cross*

Description

A pdf report will be created containing a summary of a cross object. Simultaneously the same report will be created as a tex file.

Usage

```
## S3 method for class 'cross'
report(x, ..., outfile = NULL)
```

Arguments

x	An object of class cross.
...	Further arguments passed on from other functions - not used yet.
outfile	A character string, the name and location of the output .pdf and .tex file for the report. If NULL a report with a default name will be created in the current working directory.

Examples

```
## Read the data
F2 <- qtl::read.cross(format="csv",
                      file = system.file("extdata",
                                         "F2_maize_practical3_ex2.csv",
                                         package = "RAP"),
                      genotypes = c("AA", "AB", "BB"),
```

```
                      alleles = c("A", "B"), estimate.map = FALSE)
## Not run:
## Create a report
report(F2, outfile = "./testReports/reportCross.pdf")

## End(Not run)
```

`report.FW` *Report method for class FW*

Description

A pdf report will be created containing a summary of a Finlay-Wilkinson analysis. Simultaneously the same report will be created as a tex file.

Usage

```
## S3 method for class 'FW'
report(x, sortBy = c("sens", "genMean", "mse"), ...,
  outfile = NULL)
```

Arguments

`x`	An object of class FW.
`sortBy`	A character string indicating by which variable the estimates should be sorted. Either `sens`(itivity), `genMean` (genotypic Mean) or `mse` (mean squared error).
`...`	Further arguments passed on from other functions - not used yet.
`outfile`	A character string, the name and location of the output .pdf and .tex file for the report. If `NULL` a report with a default name will be created in the current working directory.

Value

A pdf and tex report.

Examples

```
## Run Finlay-Wilkinson analysis on TDMaize.
geFW <- gxeFw(TDMaize, trait = "yld")
## Not run:
## Create a report summarizing the results.
report(geFW, outfile = "./testReports/reportFW.pdf")

## End(Not run)
```

`report.multiQTL` *Report method for class multiQTL*

Description

A pdf report will be created containing a summary of a multiQTL analysis. Simultaneously the same report will be created as a tex file.

Usage

```
## S3 method for class 'multiQTL'
report(x, ..., outfile = NULL)
```

Arguments

`x`	An object of class multiQTL.
`...`	Further arguments passed on from other functions - not used yet.
`outfile`	A character string, the name and location of the output .pdf and .tex file for the report. If `NULL` a report with a default name will be created in the current working directory.

Value

A pdf and tex report.

Examples

```
## Read the data
F2 <- qtl::read.cross(format="csv",
                      file = system.file("extdata",
                                         "F2_maize_practical3_ex2.csv",
                                         package = "RAP"),
                      genotypes = c("AA", "AB", "BB"),
                      alleles = c("A", "B"), estimate.map = FALSE)
## Perform QTL detection using simple interval mapping.
QTLDet <- QTLDetect(cross = F2, trait = "trait", type = "SIM")
## Fit a multi QTL model.
multiFit <- multiQTLFit(QTLDet)
## Not run:
## Create a pdf report summarizing results.
report(multiFit, outfile = "./testReports/reportMultiQTLFit.pdf")

## End(Not run)
```

report.QTLDet *Report method for class QTLDet*

Description

A pdf report will be created containing a summary of a QTLDet analysis. Simultaneously the same report will be created as a tex file.

Usage

```
## S3 method for class 'QTLDet'
report(x, ..., outfile = NULL)
```

Arguments

x	An object of class QTLDet.
...	Further arguments passed on from other functions - not used yet.
outfile	A character string, the name and location of the output .pdf and .tex file for the report. If NULL a report with a default name will be created in the current working directory.

Value

A pdf and tex report.

Examples

```
F2 <- qtl::read.cross(format="csv",
                      file = system.file("extdata",
                                         "F2_maize_practical3_ex2.csv",
                                         package = "RAP"),
                      genotypes = c("AA", "AB", "BB"),
                      alleles = c("A", "B"), estimate.map = FALSE)
## Perform a composite interval mapping for detecting QTLs.
QTLDet <- QTLDetect(cross = F2, trait = "trait", type = "CIM")
## Not run:
## Create a pdf report summarizing the results.
report(QTLDet, outfile = "./testReports/reportQTLDectection.pdf")

## End(Not run)
```

report.SSA *Report method for class SSA*

Description

A pdf report will be created containing a summary of the results of the fitted model. Simultaneously the same report will be created as a tex file.

Usage

```
## S3 method for class 'SSA'
report(x, ..., descending = TRUE, outfile = NULL, what = if
  (is.null(x$mFix)) "random" else "fixed")
```

Arguments

x	An object of class SSA.
...	Further arguments passed on from other functions - not used yet.
descending	Should the trait be ordered in descending order? Set to FALSE if low values of the trait indicate better performance.
outfile	A character string, the name and location of the output .pdf and .tex file for the report. If NULL a report with a default name will be created in the current working directory.
what	A character string indicating whether the model with genotype fixed or genotype random should be reported. Can be omitted if only one model has been fitted.

Value

A pdf and tex report.

Examples

```
## Fit model using lme4.
myModel1 <- STRunModel(TD = TDHeat05, design = "ibd", traits = "yield")
## Not run:
## Create a pdf report summarizing the results for the model with genotype
## as fixed factor.
report(myModel1, outfile = "./testReports/reportModelLme4.pdf",
      what = "fixed")
## Create a pdf report summarizing the results for the model with genotype
## as random factor. Order the results in ascending order.
report(myModel1, outfile = "./testReports/reportModelLme4.pdf",
      what = "random", descending = FALSE)

## End(Not run)
```

---

report.stability *Report method for class stability*

Description

A pdf report will be created containing a summary of an object of class stability. Simultaneously the same report will be created as a tex file.

Usage

```
## S3 method for class 'stability'
report(x, ..., outfile = NULL)
```

Arguments

x	An object of class stability.
...	Further arguments passed on from other functions - not used yet.
outfile	A character string, the name and location of the output .pdf and .tex file for the report. If NULL a report with a default name will be created in the current working directory.

Value

A pdf and tex report.

Examples

```
## Compute three stability measures for TDMaize.
geStab <- gxeStability(TD = TDMaize, trait = "yld")
## Not run:
## Create a .pdf report summarizing the stability measures.
report(geStab, outfile = "./testReports/reportStability.pdf")

## End(Not run)
```

---

report.varComp *Report method for class varComp*

Description

A pdf report will be created containing a summary of an object of class varComp. Simultaneously the same report will be created as a tex file.

Usage

```
## S3 method for class 'varComp'
report(x, ..., outfile = NULL)
```

Arguments

x	An object of class varComp.
...	Further arguments passed on from other functions - not used yet.
outfile	A character string, the name and location of the output .pdf and .tex file for the report. If NULL a report with a default name will be created in the current working directory.

Value

A pdf and tex report.

Examples

```
## Not run:
## Select the best variance-covariance model using asreml for modeling.
geVarComp <- gxeVarComp(TD = TDMaize, trait = "yld", engine = "asreml")
## Create a pdf report summarizing the results.
report(geVarComp, outfile = "./testReports/reportVarComp.pdf")

## End(Not run)
```

`setMeta` *Set metadata of TD objects*

Description

Function for setting metadata of a TD object for one or more trials simultaneously. Metadata can be set using a data.frame with rownames corresponding to the trials in `TD`. The data.frame should contain one or more of the following columns: trLocation, trDate, trDesign, trLat, trLong, trPlotWidth and trPlotLength. The values of the metadata of TD will be set to the values in the corresponding column in `meta`. Existing values will be overwritten, but `NA` will be ignored so setting a value to `NA` won't result in accidentally removing it.

Usage

```
setMeta(TD, meta)
```

Arguments

`TD`	An object of class TD.
`meta`	A data.frame containing metadata.

Value

The object of class TD with updated metadata.

`SSAtoCross` *Convert SSA to Cross*

Description

Convert an SSA object to a cross object from package qtl. Genotypic information should be available in a .csv file.
The only way to create an object of class cross is by importing both the phenotypic and the genotypic data from external files. Therefore the phenotypic data, either the BLUEs or the BLUPs from the fitted model are first written to a temporary file. The genotypic data has to be available in a .csv file in the correct format as well, see `genoFile` for a description of this format. These phenotypic and genotypic files are then imported into a cross object using the read.cross function in the qtl package.

Usage

```
SSAtoCross(SSA, trial = NULL, traits = NULL, what = c("BLUEs", "BLUPs"),
  genoFile, genotypes = c("A", "H", "B", "D", "C"), ...)
```

Arguments

SSA	An object of class SSA.
trial	A character string indicating the trial to be exported. If NULL and SSA contains only one trial that trial is exported.
traits	A character string containing the traits to be exported. If NULL all traits for the selected trial are exported.
what	A character string containing the statistics to be exported as phenotype in the cross object. This can be either BLUEs or BLUPs.
genoFile	A character string indicating a filename containing phenotypic data. The data should be in the format required by the qtl package. The first column should contain the individuals, starting from row 4. The following columns contain markers with in the second and third row the chromosome and position on the chromosome and in the following rows the genotypes.
genotypes	A character vector specifying the genotype codes corresponding to AA, AB, BB, not BB and not AA.
...	Further arguments to be passed to the read.cross function. See read.cross.

See Also

read.cross

Examples

```
## Run model using SpATS.
myModel <- STRunModel(TD = TDHeat05, design = "res.rowcol", traits = "yield",
                      what = "fixed")
## Create cross object with BLUEs from myModel using genotypic information
## from markers.csv in the package.
cross <- SSAtoCross(myModel, genoFile = system.file("extdata", "markers.csv",
                                                    package = "RAP"))
```

---

stability *S3 class stability*

Description

Function for creating objects of S3 class stability.
print, summary, plot and report methods are available.

Usage

```
createstability(superiority = NULL, static = NULL, wricke = NULL, trait)
```

Arguments

superiority	A data.frame containing values for the cultivar-superiority measure of Lin and Binns.
static	A data.frame containing values for Shukla's stability variance.
wricke	A data.frame containing values for Wricke's ecovalence.
trait	A character string indicating the trait that has been analyzed.

Author(s)

Bart-Jan van Rossum

See Also

plot.stability, report.stability

STExtract *Extract statistics from Fitted Models*

Description

This function extracts and calculates various results for fitted models such as BLUEs, BLUPs, unit errors and heritabilities. Note that most results can only be calculated if a model is fitted with genotype as fixed or random. This is indicated in the list below with "F" and "R"

Usage

```
STExtract(SSA, trials = names(SSA), traits = NULL, what = "all",
  keep = NULL)
```

Arguments

SSA	An object of class SSA.
trials	A character vector of trials for which the statistics should be computed. If not supplied statistics are computed for all trials that have been modelled.
traits	A character vector of traits for which the statistics should be computed. If not supplied statistics are computed for all traits that have been modelled.
what	A character vector indicating which statistics should be computed. Most statistics are available for all models, some only for models fitted using a certain engine. If this is the case this is indicated in the list with options in details. If `what = "all"` all available statistics are computed.
keep	A character vector of column(s) in the object of class TD used for modeling. These columns will be kept as output when computing fitted values, residuals, standardized residuals and rMeans. Columns can also be kept when computing (se)BLUEs and (se)BLUPs but only if the column to keep contains unique values for the modeled variables, i.e. a column repId with several different values per genotype cannot be kept.

Details

Possible options for `what` are:

F - BLUEs Best Lineair Unbiased Estimators.

F - seBLUES Standard errors of the BLUEs.

R - BLUPs Best Lineair Unbiased Predictors.

R - seBLUPs Standard errors of the BLUPs.

F - ue Unit errors - only for `lme4` and `asreml`.

R - heritability Heritability.

R - varGen Genetic variance component.

R - varErr Residual variance component - only for `lme4` and `asreml`.

R - varSpat Spatial variance components - only for `SpATS`.

F - fitted Fitted values for the model with genotype as fixed component.

F - resid Residuals for the model with genotype as fixed component.

F - stdRes Standardized residuals for the model with genotype as fixed component - only for `lme4` and `asreml`.

R - rMeans Fitted values for the model with genotype as random component.

R - ranEf Random genetic effects.

F - wald Results of the wald test - only for `lme4` and `asreml`.

F - CV Coefficient of variation - only for `lme4` and `asreml`.

F - rDf Residual degrees of freedom.

R - effDim Effective dimensions - only for `SpATS`.

F - sed Standard error of difference - only for `asreml`.

F - lsd Least significant difference - only for `asreml`.

all All available statistics.

Value

A list with per trial for which statistics have been extracted either a list of those statistics or if only one statistic is extracted a single object containing this statistic.

See Also

`STRunModel`, `STModSpATS`, `STModLme4` and `STModAsreml`

Examples

```
## Fit model using SpATS.
myModel <- STRunModel(TD = TDHeat05, design = "res.rowcol", traits = "yield")
## Extract all available statistics from the fitted model.
extr <- STExtract(myModel)
## Extract only the BLUEs from the fitted model.
BLUEs <- STExtract(myModel, what = "BLUEs")
## Extract only the BLUEs from the fitted model and keep trial as variable in
## the output.
BLUEs2 <- STExtract(myModel, what = "BLUEs", keep = "trial")
```

STRunModel *Fit Single Trial Mixed Model*

Description

Perform REML analysis given a specific experimental design. This is a wrapper function of STModSpATS, STModLme4 and STModAsreml. See details for the exact models fitted. SpATS is used as a default method when design is rowcol or res.rowcol, lme4 for other designs.

Usage

```
STRunModel(TD, trials = names(TD), design = NULL, traits,
  what = c("fixed", "random"), covariates = NULL, useCheckId = FALSE,
  trySpatial = FALSE, engine = NA, control = NULL, ...)
```

Arguments

TD	An object of class TD.
trials	A character vector specifying the trials for modeling.
design	A character string specifying the experimental design. Either "ibd" (incomplete block design), "res.ibd" (resolvable incomplete block design), "rcbd" (randomized complete block design), "rowcol" (row column design) or "res.rowcol" (resolvable row column design).
traits	A character vector specifying the traits for modeling.
what	A character vector specifying whether "genotype" should be fitted as "fixed" or "random" effect. If not specified both models are fitted.
covariates	A character vector specifying covariates to be fitted as extra fixed effects in the model.
useCheckId	Should checkId be used as a fixed effect in the model? If TRUE TD has to contain a column 'checkId'.
trySpatial	Should spatial models be tried? Spatial models can only be fitted with SpATS and asreml. If SpATS is used for modeling only spatial models can be fitted and trySpatial is always set to TRUE. If asreml is used fitting spatial models is optional.
engine	A string specifying the name of the mixed modelling engine to use, either SpATS, lme4 or asreml. For spatial models SpaTS is used as a default, for other models lme4.
control	An optional list with control parameters to be passed to the actual fitting funcions. Currently nSeg and nestDiv are valid parameters when fitting a model using SpATS. They pass a value to nseg and nest.div in PSANOVA respectively. criterion is a valid parameter when fitting a spatial model using asreml. Use this to pass a goodness-of-fit criterion for comparing different spatial models. See also in details. Other parameters are ignored.
...	Further arguments to be passed to SpATS, lme4 or asreml.

Details

The actual model fitted depends on the design. For the supported designs the following models are used:

ibd trait = genotype + *subBlock* + e

res.ibd trait = genotype + **repId** + *repId:subBlock* + e

rcbd trait = genotype + **repId** + e

rowcol trait = genotype + *rowId* + *colId* + e

res.rowcol trait = genotype + **repId** + *repId:rowId* + *repId:colId* + e

In the above models fixed effects are indicated in **bold**, random effects in *italics*. genotype is fitted as fixed or random effect depending on the value of `what`.
In case `useCheckId = TRUE` an extra fixed effect **checkId** is included in the model.
Variables in `covariates` are fitted as extra fixed effects.

When `SpATS` is used for modelling an extra spatial term is included in the model. This term is constructed using the function `PSANOVA` from the SpATS package as `PSANOVA(colCoordinates, rowCoordinates, nseg =` where `nSeg = (number of columns / 2, number of rows / 2)`. nseg and nest.div can be modified using the `control` parameter.

When `asreml` is used for modeling and `trySpatial` is `TRUE` 6 models are fitted with different random term and covariance structure. The best model is determined based on a goodness-of-fit criterion, either AIC or BIC. This can be set using the control parameter `criterion`, default is AIC. The fitted random terms depend on the structure of the data. If the design has a regular structure, i.e. all replicates appear the same amount of times in the design, the following combinations of random and spatial terms are fitted

- random = NULL, spatial = exp(rowCoordinates):colCoordinates
- random = NULL, spatial = rowCoordinates:exp(colCoordinates)
- random = NULL, spatial = iexp(rowCoordinates,colCoordinates)
- random = repId:rowId, spatial = exp(rowCoordinates):colCoordinates
- random = repId:colId, spatial = rowCoordinates:exp(colCoordinates)
- random = repId:rowId + repId:colId, spatial = iexp(rowCoordinates,colCoordinates)

If the design is not regular the following following combinations of random and spatial terms are fitted

- random = NULL, spatial = ar1(rowId):colId
- random = NULL, spatial = rowId:ar1(colId)
- random = NULL, spatial = ar1(rowId):ar1(colId)
- random = repId:rowId, spatial = ar1(rowId):colId
- random = repId:colId, spatial = rowId:ar1(colId)
- random = repId:rowId + repId:colId, spatial = ar1(rowId):ar1(colId)

If there are no replicates in the model, in the random parts above, repId is left out.

Value

An object of class SSA, a list containing per trial that has been analyzed a list of:

mRand	A list of models with fitted with genotype as random effect.
mFix	A list of models fitted with genotype as fixed effect.
TD	An object of class TD containing the data on which mRand and mFix are based.
traits	A character vector indicating the traits for which the analysis is done.
design	A character string containing the design of the trial. (see STRunModel for the possible designs).
spatial	A character string indicating the spatial part of the model. FALSE if no spatial design has been used.
engine	A character string containing the engine used for the analysis.
predicted	A character string indicating the variable that has been predicted.

References

Maria Xose Rodriguez-Alvarez, Martin P. Boer, Fred A. van Eeuwijk, Paul H.C. Eilers (2017). Correcting for spatial heterogeneity in plant breeding experiments with P-splines. Spatial Statistics https://doi.org/10.1016/j.spasta.2017.10.003

Butler, D. G., et al. (2010). Analysis of Mixed Models for S language environments: ASReml-R reference manual. Brisbane, DPI Publications

Douglas Bates, Martin Maechler, Ben Bolker, Steve Walker (2015). Fitting Linear Mixed-Effects Models Using lme4. Journal of Statistical Software, 67(1), 1-48. https://www.jstatsoft.org/article/view/v067i01/0.

See Also

STModSpATS, STModLme4, STModAsreml

Examples

```
## Fit model using lme4.
myModel1 <- STRunModel(TD = TDHeat05, design = "ibd", traits = "yield",
                       what = "fixed")
## Summarize results.
summary(myModel1)
## Create base plots of the results.
plot(myModel1)
## Not run:
## Create a pdf report summarizing results.
report(myModel1, outfile = "./testReports/reportModelLme4.pdf")

## End(Not run)
## Fit model using SpATS.
myModel2 <- STRunModel(TD = TDHeat05, design = "res.rowcol", traits = "yield",
                       what = "fixed")
summary(myModel2)
## Create spatial plots of the results.
plot(myModel2, plotType = "spatial")
## Not run:
report(myModel2, outfile = "./testReports/reportModelSpATS.pdf")
```

```
## End(Not run)

## Fit model using asreml.
## Not run:
myModel3 <- STRunModel(TD = TDHeat05, design = "res.rowcol", traits = "yield",
                       what = "fixed", engine = "asreml")
summary(myModel3)
report(myModel3, outfile = "./testReports/reportModelAsreml.pdf")

## End(Not run)
```

---

summary.SSA *Summarizing objects of class* SSA

Description

summary method for class SSA.

Usage

```
## S3 method for class 'SSA'
summary(object, trial = NULL, trait = NULL,
  digits = max(getOption("digits") - 2, 3), nBest = 20, sortBy = NULL,
  naLast = TRUE, decreasing = TRUE, ...)
```

Arguments

object	An object of class SSA.
trial	A character string indicating the trial to summarize. If trial = NULL and only one trial is modelled this trial is summarized.
trait	A character string indicating the trait to summarize. If trait = NULL and only one trait is modelled this trait is summarized.
digits	An integer indicating the number of significant digits for printing.
nBest	An integer indicating the number of the best genotypes (sorted by either BLUEs or BLUPs) to print. If NA all genotypes will be printed.
sortBy	A character string specifying how the genotypes will be sorted. Either "BLUEs", "BLUPs" or NA (i.e. no sorting).
naLast	Should missing values in the data be put last when sorting?
decreasing	Should the sort order be decreasing?
...	Further arguments passed to printCoefmat.

Examples

```
## Run a single trait analysis using SpATS.
myModel <- STRunModel(TD = TDHeat05, design = "res.rowcol", traits = "yield")
## Print a summary of the fitted model.
summary(myModel)
```

summary.TD *Summarizing objects of class* TD

Description

summary method for class TD.

Usage

```
## S3 method for class 'TD'
summary(object, ..., trial = names(object), traits,
  what = c("nObs", "nMiss", "mean", "median", "min", "max", "lowerQ",
  "upperQ", "var"))
```

Arguments

object	An object of class TD.
...	Further arguments - currently not used.
trial	A character vector specifying the trials to be plotted.
traits	A character vector specifying the traits to be summarised.
what	A character vector indicating which summary statistics should be computed. If what = "all" all available statistics are computed. Possible options are: **nVals** The number of values, i.e. non-missing + missing values. **nObs** The number of non-missing observations. **nMiss** The number of missing values. **mean** The mean. **median** The median. **min** The minimum. **max** The maximum. **range** The range (maximum - minimum). **lowerQ** The lower (25%) quantile. **upperQ** The upper (75%) quantile. **sd** The standard deviation. **seMean** The standard error of mean. **var** The variance. **seVar** The standard error of variance. **CV** The coefficient of variation. **sum** The sum. **sumSq** The sum of squares. **uncorSumSq** The uncorrected sum of squares. **skew** The skewness. **seSkew** The standard error of the skewness. **kurt** The kurtosis. **seKurt** The standard error of the kurtosis. **all** All summary statistics.

Value

A table containing the selected summary statistics.

See Also

createTD

Examples

```
## Summarize TDHeat05.
summary(object = TDHeat05, traits = "yield")
```

TD *S3 class TD*

Description

createTD
Function for creating objects of S3 class TD (Trial Data). The input data is checked and columns are renamed to default column names for ease of further computations. The columns for genotype, trial, megaEnv, year, repId, subBlock, rowId, colId and checkId are converted to factor columns, whereas rowCoordinates and colCoordinates are converted to numerical columns. One single column can be mapped to multiple defaults, e.g. one column with x coordinates can be mapped to both colId and colCoordinates.
Columns other than the default columns, e.g. traits or other covariates will be included in the output unchanged.
The input data is split by trial. So for an input data frame with three different trials in the column assigned to `trial` the output will be generated as a list with three items. In this case metadata describing the trials will be identical.
To generate a TD object with different metadata for each trial start by creating a TD object for one trial and then add new trials to it using the addTD function. Alternatively create a TD object with all data and then use setMeta to add metadata for all trials at once.

addTD
Function for adding extra trial data to an existing object of class TD. The data for the new trials will be added after the data for existing trials. It is possible to add data for an already existing trial, but this will cause multiple items in the output with identical names, which might cause problems later on in the analysis. Therefore a warning will be issued in this case.

dropTD
Function for removing data for selected trials from an existing object of class TD.

summary.TD and plot.TD methods are available.

Usage

```
createTD(data, genotype = NULL, trial = NULL, megaEnv = NULL,
  year = NULL, repId = NULL, subBlock = NULL, rowId = NULL,
  colId = NULL, rowCoordinates = NULL, colCoordinates = NULL,
  checkId = NULL, trLocation = NULL, trDate = NULL, trDesign = NULL,
```

```
  trLat = NULL, trLong = NULL, trPlotWidth = NULL, trPlotLength = NULL)

addTD(TD, data, genotype = NULL, trial = NULL, megaEnv = NULL,
  year = NULL, repId = NULL, subBlock = NULL, rowId = NULL,
  colId = NULL, rowCoordinates = NULL, colCoordinates = NULL,
  checkId = NULL, trLocation = NULL, trDate = NULL, trDesign = NULL,
  trLat = NULL, trLong = NULL, trPlotWidth = NULL, trPlotLength = NULL)

dropTD(TD, trials)
```

Arguments

data	A data.frame containing trial data with a least a column for genotype.
genotype	An optional character string indicating the column in data that contains genotypes.
trial	An optional character string indicating the column in data that contains trials.
megaEnv	An optional character string indicating the column in data that contains mega-environments as constructed by gxeMegaEnv.
year	An optional character string indicating the column in data that contains years.
repId	An optional character string indicating the column in data that contains replicates.
subBlock	An optional character string indicating the column in data that contains sub blocks.
rowId	An optional character string indicating the column in data that contains field rows.
colId	An optional character string indicating the column in data that contains field columns.
rowCoordinates	An optional character string indicating the column in data that contains the rowId coordinates used for fitting spatial models.
colCoordinates	An optional character string indicating the column in data that contains the column coordinates used for fitting spatial models.
checkId	An optional character string indicating the column in data that contains the check IDs.
trLocation	An optional character string indicating the location of the trial. This will be used for constructing default names for plots and such.
trDate	An optional date indicating the date of the trial.
trDesign	An optional character string indicating the design of the trial. Either "ibd" (incomplete-block design), "res.ibd" (resolvable incomplete-block design), "rcbd" (randomized complete block design), "rowcol" (row-column design) or "res.rowcol" (resolvable row-column design).
trLat	An optional numerical value indicating the latitude of the trial on a scale of -90 to 90.
trLong	An optional numerical value indicating the longitude of the trial on a scale of -180 to 180.
trPlotWidth	An optional positive numerical value indicating the width of the plot.
trPlotLength	An optional positive numerical value indicating the length of the plot.
TD	An object of class TD which should be modified.
trials	A character vector of trials that should be removed.

Value

An object of class TD, a list of data.frames with renamed columns and an attribute `renamedCols` containing info on which columns have been renamed. For each unique value of trial the output has a data.frame in the list with the same name as the trial. These data.frames have attributes containing the metadata for the corresponding trial. If there is no column for trial the list will contain one item named after the input data.

Author(s)

Bart-Jan van Rossum

See Also

`summary.TD`, `plot.TD`

TDHeat05 *Test data*

Description

Test data

Usage

```
TDHeat05
```

Format

An object of class `TD` (inherits from `list`) of length 1.

TDMaize *Test data maize*

Description

Test data maize

Usage

```
TDMaize
```

Format

An object of class `TD` (inherits from `list`) of length 8.

testData — *Random test data for unit testing.*

Description

Random test data for unit testing.

Usage

```
testData
```

Format

An object of class `data.frame` with 90 rows and 12 columns.

varComp — *S3 class varComp*

Description

Function for creating objects of S3 class varComp.
`print`, `summary`, `plot` and `report` methods are available.

Usage

```
createVarComp(SSA, choice, summary, vcov, criterion, engine)
```

Arguments

`SSA`	An object of class SSA, the best fitted model.
`choice`	A character string indicating the best fitted model.
`summary`	A data.frame with a summary of the fitted models.
`vcov`	The covariance matrix of the best fitted model.
`criterion`	A character string indicating the goodness-of-fit criterion used for determinening the best model.
`engine`	A character string containing the engine used for the analysis.

Author(s)

Bart-Jan van Rossum

See Also

`plot.varComp`, `report.varComp`

Index