Package 'asremlPlus'

August 12, 2018

Version 4.1-03

Date 2018-08-12

Title Augments the Use of 'ASReml-R' and 'ASReml4-R' in Fitting Mixed Models

Author Chris Brien <Chris.Brien@unisa.edu.au>.

Maintainer Chris Brien <Chris.Brien@unisa.edu.au>

Depends R (>= 3.1.0)

Imports dae, ggplot2, stats, methods, utils, reshape, plyr, dplyr, stringr, RColorBrewer, grDevices, foreach, parallel, doParallel

Suggests testthat, lattice

Enhances asreml

SystemRequirements asreml-R 2.x

Description Assists in automating the testing of terms in mixed models when 'asreml' is used to fit the models. The content falls into the following natural groupings: (i) Data, (ii) Object manipulation functions, (iii) Model modification functions, (iv) Model testing functions, (v) Model diagnostics functions, (vi) Prediction production and presentation functions, (vii) Response transformation functions, and (viii) Miscellaneous functions. A history of the fitting of a sequence of models is kept in a data frame. Procedures are available for choosing models that conform to the hierarchy or marginality principle and for displaying predictions for significant terms in tables and graphs. The packages 'asreml' and 'asreml4' provide a computationally efficient algorithm for fitting mixed models using Residual Maximum Likelihood. They are commercial packages that can be purchased from 'VSNi' <http://www.vsni.co.uk/> as 'asreml-R', who will supply a zip file for local installation/updating.

License MIT + file LICENSE

URL http://chris.brien.name

NeedsCompilation no

R topics documented:

1

Index   **102**

| asremlPlus-package | *Augments the Use of 'ASReml-R' and 'ASReml4-R' in Fitting Mixed Models* |

Description

Assists in automating the testing of terms in mixed models when 'asreml' is used to fit the models. The content falls into the following natural groupings: (i) Data, (ii) Object manipulation functions, (iii) Model modification functions, (iv) Model testing functions, (v) Model diagnostics functions, (vi) Prediction production and presentation functions, (vii) Response transformation functions, and (viii) Miscellaneous functions. A history of the fitting of a sequence of models is kept in a data frame. Procedures are available for choosing models that conform to the hierarchy or marginality principle and for displaying predictions for significant terms in tables and graphs. The packages 'asreml' and 'asreml4' provide a computationally efficient algorithm for fitting mixed models using Residual Maximum Likelihood. They are commercial packages that can be purchased from 'VSNi' <http://www.vsni.co.uk/> as 'asreml-R', who will supply a zip file for local installation/updating.

Version: 4.1-03

Date: 2018-08-12

Index

(i) Data

Wheat.dat	Data for an experiment to investigate 25 varieties of wheat.
WaterRunoff.dat	Data for an experiment to investigate the quality of water runoff over time

(ii) Object manipulation

as.alldiffs	Forms an object of S3-class 'alldiffs' that stores the predictions for a model fitted using asreml, along with statistics for all pairwise differences.
asrtests	Forms an object of S3-class 'asrtests' that stores (i) a fitted asreml object, (ii) a pseudo-anova table for the fixed and (iii) a history of changes and hypthesis testing used in obtaining the model.
facCombine.alldiffs	Combines several factors into one in the components of an alldiffs object.
print.alldiffs	Prints the values in an 'alldiffs' object in a nice format.
print.asrtests	Prints the values in an 'asrtests' object.
sort.alldiffs	Sorts the components of an 'alldiffs' object according to the predicted values associated with a factor.
subset.alldiffs	Subsets the components in an alldiffs object according to the supplied condition.

(iii) Model modification

changeTerms.asrtests	Adds and drops the specified sets of terms from one or both of the fixed or random model and/or replaces the

	residual (rcov) model with a new model.
newfit.asreml	Refits an asreml model with modified model formula using either a call to 'update.asreml' or a direct call to 'asreml'.
reparamSigDevn.asrtests	Reparamterizes each random (deviations) term involving 'devn.fac' to a fixed term and ensures that the same term, with 'trend.num' replacing 'devn.fac', is included if any other term with 'trend.num' is included in 'terms'.
rmboundary.asrtests	Removes any boundary or singular variance components from the fit stored in 'asreml.obj' and records their removal in a data.frame.
setvarianceterms.call	Allows the seting of bounds and initial values for terms in the 'random' and 'residual' arguments of an 'asreml' call.

(iv) Model testing

chooseModel.asrtests	Determines the set of significant terms taking into account hierarchy or marginality relations and records the tests performed in a data.frame.
infoCriteria.asreml	Computes AIC and BIC for a model.
recalcWaldTab.asrtests	Recalculates the denDF, F.inc and P values for a table of Wald test statistics obtained using 'wald.asreml'.
REMLRT.asreml	Performs a REML ratio test.
bootREMLRT.asreml	Performs a REML ratio test using the parametric bootstrap.
testranfix.asrtests	Tests for a single fixed or random term in model fitted using 'asreml' and records the result in a data.frame.
testresidual.asrtests	Fits a new residual formula using 'asreml', tests whether the change is significant and records the result in a data.frame.
testswapran.asrtests	Tests, using a REMLRT, the significance of the difference between the current random model and one in which oldterms are dropped and newterms are added. The result is recorded in a data.frame.

(v) Model diagnostics and simulation

plotVariofaces	Plots empirical variogram faces, including envelopes, from supplied residuals as described by Stefanova, Smith & Cullis (2009).
variofaces.asreml	Calculates and plots empirical variogram faces, including envelopes, as described by Stefanova, Smith & Cullis (2009).
estimateV.asreml	Forms the estimated variance, random or residual matrix for the observations from the variance parameter estimates.
simulate.asreml	Produce sets of simulated data from a multivariate normal distribtion and save quantites related to the simulated data.

(vi) Prediction production and presentation

allDifferences.data.frame	Using supplied predictions, forms all pairwise differences between a set of predictions, their standard errors and p-values for the differences.
linTransform.alldiffs	Calculates a linear transformation of the predictions stored in an alldiffs object.
plotPredictions.data.frame	Plots the predictions for a term, possibly with error bars.
plotPvalues.alldiffs	Plots the p-values in the p.differences components of an alldiffs object as a heat map.
plotPvalues.data.frame	Plots the p-values in data.frame as a heat map.
predictPlus.asreml	Forms the predictions and associated statistics for a term, using an asreml object and a wald.tab and taking into account that a numeric vector and a factor having parallel values may occur in the model. It stores the results in an object of class 'alldifffs' and may print the results. It can be when there are not parallel values.
predictPresent.asreml	Forms the predictions for each of one or more terms and presents them in tables and/or graphs.
recalcLSD.alldiffs	Adds or recalculates the LSD component of an alldiffs object.
redoErrorIntervals.alldiffs	Adds or replaces the error intervals stored in the prediction component of an alldiffs.object.
reorderClassify.alldiffs	Reorders the components in an alldiffs object according to a new classify.
sort.alldiffs	Sorts the components in an alldiffs object according to the predicted values associated with a factor.
subset.alldiffs	Subsets the components in an alldiffs object according to the supplied condition.

(vii) Response transformation

angular	Applies the angular transformation to proportions.
angular.mod	Applies the modified angular transformation to a vector of counts.
powerTransform	Performs a combination of a linear and a power transformation on a variable. The transformed variable is stored in the 'data.frame data'.

(viii) Miscellaneous

getASRemlVersionLoaded	Finds the version of asreml that is loaded and returns the initial characters in version.
loadASRemlVersion	Ensures that a specific version of asreml is loaded.
num.recode	Recodes the unique values of a vector using the values in a new vector.
permute.square	Permutes the rows and columns of a square matrix.
permute.to.zero.lowertri	Permutes a square matrix until all the lower triangular elements are zero.

The functions whose names end in 'asrtests', which are most of the model functions, utilize an `asrtests` object that stores: (i) the currently fitted model in `asreml.obj`, (ii) the table of test statistics for the fixed effects in `wald.tab`, and (iii) a data frame that contains a history of the changes made to the model in `test.summary`.

Author(s)

Chris Brien <Chris.Brien@unisa.edu.au>.

Maintainer: Chris Brien <Chris.Brien@unisa.edu.au>

References

Butler, D. G., Cullis, B. R., Gilmour, A. R., Gogel, B. J. and Thompson, R. (2017). *ASReml-R User Guide Version 4.* VSN International Ltd, <http://www.vsni.co.uk/software/asreml/>.

Stefanova, K. T., Smith, A. B. & Cullis, B. R. (2009) Enhanced diagnostics for the spatial analysis of field trials. *Journal of Agricultural, Biological, and Environmental Statistics*, **14**, 392–410.

See Also

`asreml`

Examples

```
## Not run:
## Analyse wheat dat using asreml4 and asremlPlus
## Set up for analysis
library(dae)
library(asreml4)
library(asremlPlus)
## use ?Wheat.dat for data set details
data(Wheat.dat)

# Fit initial model
current.asr <- asreml(yield ~ Rep + WithinColPairs + Variety,
                      random = ~ Row + Column + units,
                      residual = ~ ar1(Row):ar1(Column),
                      data=Wheat.dat)
summary(current.asr)

# Load current fit into an asrtests object
current.asrt <- asrtests(current.asr, NULL, NULL)

# Check for and remove any boundary terms
current.asrt <- rmboundary(current.asrt)

# Check term for within Column pairs
current.asrt <- testranfix(current.asrt, "WithinColPairs", drop.fix.ns=TRUE)

# Test nugget term
current.asrt <- testranfix(current.asrt, "units", positive=TRUE)

# Test Row autocorrelation
current.asrt <- testresidual(current.asrt, "~ Row:ar1(Column)",
                             label="Row autocorrelation", simpler=TRUE)
```

```
# Test Col autocorrelation (depends on whether Row autocorrelation retained)
k <- match("Row autocorrelation", current.asrt$test.summary$terms)
p <- current.asrt$test.summary$p
{ if (p[k] <= 0.05)
    current.asrt <- testresidual(current.asrt, "~ ar1(Row):Column",
                                 label="Col autocorrelation", simpler=TRUE,
                                 update=FALSE)
    else
      current.asrt <- testresidual(current.asrt, "~ Row:Column",
                                   label="Col autocorrelation", simpler=TRUE,
                                   update=FALSE)
}
print(current.asrt)
info <- infoCriteria(current.asrt$asreml.obj)

# Get current fitted asreml object
current.asr <- current.asrt$asreml.obj
current.asr <- update(current.asr, aom=TRUE)


# Do residuals-versus-fitted values plot
plot(fitted(current.asr),residuals(current.asr))

#Produce variogram and variogram faces plot (Stefanaova et al, 2009)
plot.varioGram(varioGram(current.asr))
faces <- variofaces(current.asr, V=V, maxiter=50, units="addtores")

#Get Variety predictions, sorted in increasing order for the predicted values,
#and all pairwise prediction differences and p-values. Plot the latter.
Var.diffs <- predictPlus(classify = "Variety",
                         asreml.obj=current.asr,
                         error.intervals="halfLeast",
                         wald.tab=current.asrt$wald.tab,
                         tables = "predictions",
                         sortFactor = "Variety")
print(Var.diffs, which = c("differences", "p.differences"))
plotPvalues(Var.diffs)

## End(Not run)
```

---

allDifferences.data.frame

Using supplied predictions, forms all pairwise differences between a set of predictions, their standard errors and p-values for the differences.

Description

Uses predictions and standard errors of pairwise differences from an alldiffs object to form, for those components not already present, (i) a table of all pairwise differences of the predictions in an alldiffs object, (ii) the p-values of each pairwise difference, and (iii) the minimum, mean and maximum LSD values. Predictions that are aliased (or nonestimable) are removed from the predictions component of the alldiffs object and standard errors of differences involving them are removed from the sed component.

Also, the rows of `predictions` component are ordered so that they are in standard order for the variables in the `classify`. That is, the values of the last variable change with every row, those of the second-last variable only change after all the values of the last variable have been traversed; in general, the values of a variable are the same for all the combinations of the values to the variables to its right in the `classify`. In addition, if necessary, the order of the columns of the variables in the `predictions` component are changed to match their order in the `classify`. The `sortFactor` or `sortOrder` arguments can be used to order of the values for the `classify` variables, which is achieved using `sort.alldiffs`.

Each p-value is computed as the probability of a t-statistic as large as or larger than the absolute value of the observed difference divided by its standard error. The p-values are stored in the `p.differences` component. The degrees of freedom of the t-distribution is the degrees of freedom stored in the `tdf` attribute of the `alldiffs` object. This t-distibution is also used in calculating the LSD statistics stored in the `alldiffs` object.

Usage

```
## S3 method for class 'data.frame'
allDifferences(predictions, classify, vcov = NULL,
               differences = NULL, p.differences = NULL, sed = NULL,
               LSD = NULL, meanLSD.type = "overall", LSDby = NULL,
               backtransforms = NULL,
               response = NULL, response.title = NULL,
               term = NULL, tdf = NULL,
               x.num = NULL, x.fac = NULL,
               level.length = NA,
               pairwise = TRUE, alpha = 0.05,
               inestimable.rm = TRUE,
               sortFactor = NULL, sortWithinVals = NULL,
               sortOrder = NULL, decreasing = FALSE, ...)
```

Arguments

predictions   a `data.frame` containing the predicted values that is consistent with an object of class `asremlPredict` such as is stored in the `pvals` component of the `prediction` component of the value produced by `predict.asreml`. That is, in addition to variables classifying the predictions, it will include columns named `predicted.value`, `standard.error` and `est.status`; each row contains a single predicted value. It may also contain columns for the lower and upper confidence limits for the predictions. If `LSD` is not `NULL` then the mean LSD will be added as an attribute named `meanLSD`, it being calculated as the square root of the mean of the variances of pairwise differences.

classify   a `character` string giving the variables that define the margins of the multiway table that has been predicted. Multiway tables are specified by forming an interaction type term from the classifying variables, that is, separating the variable names with the `:` operator.

vcov   a `matrix` containing the variance matrix of the predictions; it is used in computing the variance of linear transformations of the predictions.

differences   a `matrix` containing all pairwise differences between the predictions; it should have the same number of rows and columns as there are rows in `predictions`.

p.differences   a `matrix` containing p-values for all pairwise differences between the predictions; each p-value is computed as the probability of a t-statistic as large as or

	larger than the observed difference divided by its standard error. The degrees of freedom of the t distribution for computing it are computed as the denominator degrees of freedom of the F value for the fixed term, if available; otherwise, the degrees of freedom stored in the attribute tdf are used; the matrix should be of the same size as that for differences.
sed	a matrix containing the standard errors of all pairwise differences between the predictions; they are used in computing the p-values.
LSD	a data.frame containing the mean, minimum and maximum LSD for determining the significance of pairwise differences, the mean LSD being calculated using the square root of the mean of the variances of pairwise differences.
meanLSD.type	A character string determining whether the mean LSD stored is (i) the overall mean, based on the square root of the mean of the variances of all pairwise variances, (ii) the mean for each factor.combination of the factors specified by LSDby, which is based on the square root of the mean of the variances for all pairwise differences for each factor combination, or (iii) the per.prediction mean, based, for each prediction, on the square root of the mean of the variances for all pairwise differences involving that prediction.
LSDby	A character (vector) of factor names, being the names of the factors for each of whose levels combinations a mean LSD, minLSD and max LSD is stored in the LSD component of the alldiffs object when meanLSD.type is factor.combinatons.
backtransforms	a data.frame containing the backtransformed values of the predicted values that is consistent with an object of class asremlPredict such as is stored in the pvals component of the prediction component of the value produced by predict.asreml. That is, in addition to variables classifying the predictions, it will include columns named backtransformed.predictions and est.status; it may also contain columns for the lower and upper confidence limits; each row contains a single predicted value.
response	a character specifying the response variable for the predictions. It is stored as an attribute to the alldiffs object.
response.title	a character specifying the title for the response variable for the predictions. It is stored as an attribute to the alldiffs object.
term	a character string giving the variables that define the term that was fitted using asreml and that corresponds to classify. It is often the same as classify. It is stored as an attribute to the alldiffs object.
tdf	an integer specifying the degrees of freedom of the standard error. It is used as the degrees of freedom for the t-distribution on which p-values and confidence intervals are based. It is stored as an attribute to the alldiffs object.
x.num	A character string giving the name of the numeric covariate that corresponds to x.fac, is potentially included in terms in the fitted model and which corresponds to the x-axis variable. It should have the same number of unique values as the number of levels in x.fac.
x.fac	A character string giving the name of the factor that corresponds to x.num, is potentially included in terms in the fitted model and which corresponds to the x-axis variable. It should have the same number of levels as the number of unique values in x.num. The levels of x.fac must be in the order in which they are to be plotted - if they are dates, then they should be in the form yyyymmdd, which can be achieved using as.Date. However, the levels can be non-numeric in nature, provided that x.num is also set.

level.length	The maximum number of characters from the the levels of factors to use in the row and column labels of the tables of pairwise differences and their p-values and standard errors.
pairwise	A logical indicating whether all pairwise differences of the predictions and their standard errors and p-values are to be computed and stored. If FALSE, the components differences and p.differences will be NULL in the returned alldiffs object.
alpha	The significance level for an LSD to compare a pair of predictions.
inestimable.rm	A logical indicating whether rows for predictions that are not estimable are to be removed from the components of the alldiffs object.
sortFactor	A character containing the name of the factor that indexes the set of predicted values that determines the sorting of the alldiffs components by sort.alldiffs. If NULL then sorting is not carried out. If there is more than one variable in the classify term then sortFactor is sorted for the predicted values within each combination of the values of the sortWithin variables: the classify variables, excluding the sortFactor. There should be only one predicted value for each unique value of sortFactor within each set defined by a combination of the values of the sortWithin variables.
sortWithinVals	A list with a component named for each factor and numeric that is a classify variable for the predictions, excluding sortFactor. Each component should contain a single value that is a value of the variable. The combination of this set of values will be used to define a subset of the predicted values whose order will define the order of sortFactor to be used for all combinations of the sortWithinVals variables. If sortWithinVals is NULL then the first value of each sortWithin variable in predictions component is used to define sortWithinVals. If there is only one variable in the classify then sortWithinVals is ignored.
sortOrder	A character vector whose length is the same as the number of levels for sortFactor in the predictions component of the alldiffs object. It specifies the desired order of the levels in the reordered components of the alldiffs object. The argument sortWithinVals is ignored. The following creates a sortOrder vector levs for factor f based on the values in x: levs <- levels(f)[order(x)].
decreasing	A logical passed to order that detemines whether the order for sorting the alldiffs components is for increasing or decreasing magnitude of the predicted values.
...	provision for passsing arguments to functions called internally - not used at present.

Value

An alldiffs object that is a list with components predictions containing the predictions and their standard errors, vcov containing the variance matrix of the predictions, differences containing all pairwise differences between the predictions, p.differences containing p-values for all pairwise differences between the predictions, sed containing the standard errors of all pairwise differences between the predictions, and an LSD containing the mean, minimum and maximum LSDs.

The name of the response, the response.title, the term, the classify, tdf, sortFactor and the sortOrder will be set as attributes to the object. Note that the classify in an alldiffs object is based on the variables indexing the predictions, which may differ from the classify used to obtain the original predictions (for example, when the alldiffs objects stores a linear transformation of predictions.

See predictPlus.asreml for more information.

Author(s)

Chris Brien

See Also

asremlPlus-package, as.alldiffs, sort.alldiffs, subset.alldiffs, print.alldiffs, reorderClassify.alldiffs, redoErrorIntervals.alldiffs, recalcLSD.alldiffs, plotPredictions.data.frame, predictPlus.asreml, predictPresent.asreml

Examples

```
## Not run:
  data(Oats.dat)
  m1.asr <- asreml(Yield ~ Nitrogen*Variety,
                   random=~Blocks/Wplots,
                   data=Oats.dat)
  current.asrt <- asrtests(m1.asr)
  Var.pred <- asreml:::predict.asreml(m1.asr, classify="Nitrogen:Variety",
                                      sed=TRUE)$predictions
  wald.tab <-  current.asrt$wald.tab
  den.df <- wald.tab[match("Variety", rownames(wald.tab)), "denDF"]

  ## Order the Varieties in decreasing order for the values of the predictions
  ## in the first N level
  Var.diffs <- allDifferences(predictions = Var.pred$pvals, classify = "Variety",
                              sed = Var.pred$sed, tdf = den.df,
                              sortFactor = "Variety", decreasing = TRUE)
  print.alldiffs(Var.diffs, which="differences")

  ## Change the order of the factors in the alldiffs object and reorder components
  Var.reord.diffs <- allDifferences(predictions = Var.pred$pvals,
                             classify = "Variety:Nitrogen",
                             sed = Var.pred$sed, tdf = den.df)
  print.alldiffs(Var.diffs, which="predictions")


## End(Not run)
```

| angular | *Applies the angular transformation to proportions.* |

Description

Applies the angular transformation to numeric values. It is given by $\sin^{-1}(\sqrt{proportions})$

Usage

```
angular(proportions, n)
```

Arguments

| proportions | The proportions. |
| n | The divisor(s) for each proportion |

Value

A numeric.

Author(s)

Chris Brien

See Also

angular.mod, powerTransform.

Examples

```
n <-25
y <- rbinom(10, n, 0.5)
y <- c(y,0,n)
p <- y/n
p.ang <- angular(p, n)
```

---

angular.mod *Applies the modified angular transformation to a vector of counts.*

Description

Applies the angular transformation to a vector of counts. A modified transformation is used that is appropriate when N < 50 and the proportion is not between 0.3 and 0.7. The transformation is given by $\sin^{-1} \frac{count+0.375}{n+0.75}$ arcsin(sqrt((count + 0.375) / (n + 0.75))).

Usage

```
angular.mod(count, n)
```

Arguments

count	The numeric vector of counts.
n	The number(s) of observations from which the count(s) were obtained.

Value

A numeric vector.

Author(s)

Chris Brien

See Also

angular, powerTransform.

Examples

```
n <-25
y <- rbinom(10, n, 0.5)
y <- c(y,0,n)
p.ang.mod <- angular.mod(y, n)
```

---

as.alldiffs | *Forms an object of S3-class* alldiffs *that stores the predictions for a model fitted using asreml, along with supplied statistics for all pairwise differences.*

Description

Creates an object of S3-class alldiffs that consists of a list containing the following components: predictions, vcov, differences, p.differences, sed, LSD and backtransforms. Predictions must be supplied to the function while the others will be set only if they are supplied; those not supplied are set to NULL. It also has attributes response, response.title, term, classify, tdf, sortFactor and sortOrder. which will be set to the values supplied or NULL if none are supplied.

Usage

```
as.alldiffs(predictions, vcov = NULL, differences = NULL,
            p.differences = NULL, sed = NULL, LSD = NULL,
            backtransforms = NULL,
            response = NULL, response.title = NULL,
            term = NULL, classify = NULL, tdf = NULL,
            sortFactor = NULL, sortOrder = NULL)
```

Arguments

predictions
: a data.frame containing the predicted values that is consistent with an object of class asremlPredict such as is stored in the pvals component of the prediction component of the value produced by predict.asreml. That is, in addition to variables classifying the predictions, it will include columns named predicted.value, standard.error and est.status; each row contains a single predicted value. Note that the names standard.error and est.status have been changed to std.error and status in asreml4; if the new names are present in predictions, they will be returned to the previous names. It may also contain columns for the lower and upper confidence limits for the predictions. If LSD is not NULL then the mean LSD will be added as an attribute named meanLSD, it being calculated using the square root of the mean of the variances of pairwise differences.

differences
: a matrix containing all pairwise differences between the predictions; it should have the same number of rows and columns as there are rows in predictions.

p.differences
: a matrix containing p-values for all pairwise differences between the predictions; each p-value is computed as the probability of a t-statistic as large as or larger than the observed difference divided by its standard error. The degrees of freedom of the t distribution for computing it are computed as the denominator degrees of freedom of the F value for the fixed term, if available; otherwise, the degrees of freedom stored in the attribute tdf are used; the matrix should be of the same size as that for differences.

sed	a `matrix` containing the standard errors of all pairwise differences between the predictions; they are used in computing the p-values.
vcov	a `matrix` containing the variance matrix of the predictions; it is used in computing the variance of linear transformations of the predictions.
LSD	a `data.frame` containing the mean, minimum and maximum LSD for determining the significance of pairwise differences, the mean LSD being calculated using the square root of the mean of the variances of pairwise differences. If `factor.combination` was specified for `meanLSD.type` when the LSDs were being calculated, then LSD contains an LSD for each `factor.combination` of the [factor](#)s specified by LSDby; each LSD is the square root of the mean of the variances for all pairwise differences for each factor combination.
backtransforms	a `data.frame` containing the backtransformed values of the predicted values that is consistent with an object of class asremlPredict such as is stored in the pvals component of the `prediction` component of the value produced by `predict.asreml`. That is, in addition to variables classifying the predictions, it will include columns named `backtransformed.predictions` and `est.status`; it may also contain columns for the lower and upper confidence limits; each row contains a single predicted value.
response	a `character` specifying the response variable for the predictions. It is stored as an attribute to the `alldiffs` object.
response.title	a `character` specifying the title for the response variable for the predictions. It is stored as an attribute to the `alldiffs` object.
term	a character string giving the variables that define the term that was fitted using `asreml` and that corresponds to `classify`. It is often the same as `classify`. It is stored as an attribute to the `alldiffs` object.
classify	a character string giving the variables that define the margins of the multiway table used in the prediction. Multiway tables are specified by forming an interaction type term from the classifying variables, that is, separating the variable names with the : operator. It is stored as an attribute to the `alldiffs` object.
tdf	an `integer` specifying the degrees of freedom of the standard error. It is used as the degrees of freedom for the t-distribution on which p-values and confidence intervals are based. It is stored as an attribute to the `alldiffs` object.
sortFactor	A `character` containing the name of the `factor` that indexes the set of predicted values that determined the sorting of the components.
sortOrder	A `character` vector that is the same length as the number of levels for `sortFactor` in the `predictions` component of the `alldiffs` object. It specifies the order of the levels in the reordered components of the `alldiffs` object.
	The following creates a `sortOrder` vector levs for factor f based on the values in x: levs <- levels(f)[order(x)].

Value

An object of S3-class `alldiffs` that consists of a `list` containing the following components as described in the arguments above: predictions, vcov, differences, p.differences, sed, LSD and backtransforms and with attributes names, class, response, response.title, term, classify, tdf, meanLSD, sortFactor and sortOrder. The last two attributes are also set by [sort.alldiffs](#).

The name of the response, the response.title, the term, the classify, tdf, sortFactor and the sortOrder will be set as attributes to the object. Note that the classify in an alldiffs object is based on the variables indexing the predictions, which may differ from the classify

used to obtain the original predictions (for example, when the alldiffs objects stores a linear transformation of predictions.

See `predictPlus.asreml` for more information.

Author(s)

Chris Brien

See Also

`asremlPlus-package`, `print.alldiffs`, `sort.alldiffs`, `subset.alldiffs`, `allDifferences.data.frame`, `reorderClassify.alldiffs`, `redoErrorIntervals.alldiffs`, `recalcLSD.alldiffs`, `predictPlus.asreml`, `plotPredictions.data.frame`, `predictPresent.asreml`

Examples

```
## Not run:
  Var.pred <- predict(current.asr, classify="Variety", sed=TRUE)$predictions
  wald.tab <-  current.asrt$wald.tab
  den.df <- wald.tab[match("Variety", rownames(wald.tab)), "denDF"]
  Var.diffs <- as.alldiffs(predictions = Var.pred$pvals,
                           sed = Var.pred$sed,
                           classify = "Variety", response = "Yield", tdf = den.df)

## End(Not run)
```

---

asremlPlus-deprecated   *Deprecated Functions in the Package asremlPlus*

Description

These functions have been renamed and deprecated in `asremlPlus`:

1. addrm.terms.asreml and addrm.terms.asrtests -> `changeTerms.asrtests`,
2. alldiffs -> `as.alldiffs`,
3. choose.model.asreml and choose.model.asrtests -> `chooseModel.asrtests`,
4. info.crit and info.crit.asreml -> `infoCriteria.asreml`,
5. newrcov.asrtests -> `changeTerms.asrtests`,
6. plotvariofaces.asreml -> `plotVariofaces.data.frame`,
7. power.transform -> `powerTransform`,
8. predictiondiffs.asreml -> `allDifferences.data.frame`,
9. predictionplot.asreml -> `plotPredictions.data.frame`,
10. predictparallel.asreml -> `predictPlus.asreml`,
11. pred.present.asreml -> `predictPresent.asreml`,
12. recalc.wald.tab.asreml and recalc.wald.tab.asrtests -> `recalcWaldTab.asrtests`,
13. reml.lrt and reml.lrt.asreml -> `REMLRT.asreml`,
14. rmboundary.asreml -> `rmboundary.asrtests`,

15. setvarianceterms.asreml -> setvarianceterms.call,

16. sig.devn.reparam.asreml and sig.devn.reparam.asrtests -> reparamSigDevn.asrtests,

17. testranfix.asreml -> testranfix.asrtests,

18. testrcov.asreml and testrcov.asrtests -> testresidual.asrtests,

19. testswapran.asreml -> testswapran.asrtests

Usage

```
addrm.terms.asreml(...)
addrm.terms.asrtests(...)
alldiffs(...)
choose.model.asreml(...)
choose.model.asrtests(...)
info.crit(...)
info.crit.asreml(...)
newrcov.asrtests(...)
plotvariofaces.asreml(...)
power.transform(...)
predictiondiffs.asreml(...)
predictionplot.asreml(...)
predictparallel.asreml(...)
pred.present.asreml(...)
recalc.wald.tab.asreml(...)
recalc.wald.tab.asrtests(...)
reml.lrt(...)
reml.lrt.asreml(...)
## S3 method for class 'asreml'
rmboundary(...)
setvarianceterms.asreml(...)
sig.devn.reparam.asreml(...)
sig.devn.reparam.asrtests(...)
testranfix.asreml(...)
testrcov.asreml(...)
testrcov.asrtests(...)
## S3 method for class 'asreml'
testswapran(...)
```

Arguments

...   absorbs arguments passed from the old functions of the style foo.bar().

Author(s)

Chris Brien

---

asremlPlusTips   *The randomly-presented, startup tips.*

Description

The intermittent, randomly-presented, startup tips.

Startup tips

Need help? The manual is in the doc subdirectory of the package's install directory.

Find out what has changed in asremlPlus: enter news(package = 'asremlPlus').

Need help getting started? Look at the example in ¿asremlPlus-package'.

To avoid start-up message that ASReml-R is needed, load asreml before asremlPlus.

Use suppressPackageStartupMessages() to eliminate all package startup messages.

To see all the intermittent, randomly-presented, startup tips enter ?asremlPlusTips.

To install the latest version: enter devtools::install_github('briencj/asremlPlus')

For versions between CRAN releases (and more) go to http://chris.brien.name/rpackages.

Author(s)

Chris Brien

---

asrtests	*Forms an object of S3-class* asrtests *that stores (i) a fitted asreml object, (ii) a pseudo-anova table for the fixed and (iii) a history of changes and hypthesis testing used in obtaining the model.*

Description

An object of S3-class asrtests consists of a list containing:

1. asreml.obj: an asreml object containing the fit of the model;

2. wald.tab: a data.frame containing a pseudo-anova table for the fixed terms produced by wald.asreml, which will be called if wald.tab is NULL;

3. test.summary: a data.frame with columns term, DF, denDF, p and action. A row is added to it for each term that is dropped, added or tested or a note that several terms have been added or removed. A row contains the name of the term, the DF, the p-value and the action taken. Possible codes are: Dropped, Retained, Swapped, Unswapped, Significant, Nonsignificant, Absent, Added, Removed and Boundary. If the changed model did not converge, Unconverged will be added to the code. Note that the logical asreml.obj$converge also reflects whether there is convergence.

A call to asrtests with test.summary = NULL re-initializes the test.summary data.frame.

If there is no wald.tab, wald.asreml is called. In all cases, recalcWaldTab is called and any changes made as specified by the the recalcWaldTab arguments supplied via ....

Usage

```
asrtests(asreml.obj, wald.tab = NULL, test.summary = NULL,
         denDF = "numeric", ...)
```

Arguments

asreml.obj	an asreml object for a fitted model.
wald.tab	a data.frame containing a pseudo-anova table for the fixed terms produced by wald.asreml; it should have 4 columns. Sometimes wald.asreml returns a data.frame and at other times a list. For example, it may return a list when denDF is used. In this case, the Wald component of the list is to be extracted and stored. It is noted that, as of asreml version 4, wald.asreml has a codekenadj argument.
test.summary	a data.frame with columns term, DF, denDF, p and action containing the results of previous hypothesis tests.
denDF	Specifies the enthod to use in computing approximate denominator degrees of freedom when wald.asreml is called. Can be none to suppress the computations, numeric for numerical methods, algebraic for algebraic methods or default, the default, to autommatically choose numeric or algebraic computations depending on problem size. The denominator degrees of freedom are calculated according to Kenward and Roger (1997) for fixed terms in the dense part of the model.
...	further arguments passed to wald.asreml and recalcWaldTab.

Value

An object of S3-class asrtests.

Author(s)

Chris Brien

See Also

asremlPlus-package, recalcWaldTab, testranfix.asrtests, chooseModel.asrtests, rmboundary.asrtests, reparamSigDevn.asrtests

Examples

```
## Not run:
data(Wheat.dat)

# Fit initial model
current.asr <- asreml(yield ~ Rep + WithinColPairs + Variety,
                      random = ~ Row + Column + units,
                      residual = ~ ar1(Row):ar1(Column),
                      data=Wheat.dat)

# Load current fit into an asrtests object
current.asrt <- asrtests(current.asr, NULL, NULL)

# Check for and remove any boundary terms
current.asrt <- rmboundary(current.asrt)

## End(Not run)
```

---

bootREMLRT.asreml   *Uses the parametric bootstrap to calculate the p-value for a REML ratio test to compare two models.*

Description

Extracts the REML log likelihood for two asreml objects and forms the observed REML ratio statistic. It assumes that the second asreml object is the result of fitting a model that is a reduced version of the model for the first object and is considered to the null model. Using the mean and V, nboot bootstrap samples of simulated response values are generated in parallel; that is, ncores cores are used and each is used to generate and analyse a sample. The full and reduced models are fitted to the data and if either analysis fails to converge another sample is generated and analysed using the current core, with a maximum of max.retries attempts to obtain a sample that converges for both analysis. Thus the maximum number of data sets that will be generated is nboot * max.retries. If a bootstrap sample converges for both analyses, the REML ratio test statistic is formed for it. The p-value is then calculated as $(k+1)/(b+1)$ where k is the number of simulated ratio test statistics greater than the observed test statistic and s is the number of bootstrap samples that were returned.

The function checks that the models do not differ in either their fixed or sparse models. It also check the difference in the number of variance parameters between the two fits to the models, taking into account the bound.exclusions.

Usage

```
## S3 method for class 'asreml'
bootREMLRT(h0.asreml.obj, h1.asreml.obj,
           nboot = 100, max.retries = 5, seed = NULL,
           means=NULL, V = NULL, extra.matrix = NULL, ignore.terms = NULL,
           fixed.spline.terms = NULL,
           bound.exclusions = c("F","B","S","C"),
           tolerance = 1E-10, update = TRUE, trace = FALSE,
           ncores = detectCores(), ...)
```

Arguments

h0.asreml.obj	asreml object containing the fit under the model for the null hypothesis.
h1.asreml.obj	asreml object containing the fit under the model for the alternative hypothesis.
nboot	The number of bootstrap samples to be generated.
max.retries	The maximum number of attempts to generate a sample whose analyses converge for both models.
seed	A single value, interpreted as an integer, that specifies the starting value of the random number generator. The "L'Ecuyer-CMRG" random generator is used and nextRNGStream is used to seed each core from the original seed.
means	The vector of means to be used in generating simulated bootstrap samples. If it is NULL, the fitted values based on object are used. It must be the same length as the response variable for object.
V	The fitted variance matrix, i.e. having the pattern and values that conform to the model fit stored in the supplied object. If it is NULL, estimateV.asreml is used to estimate the variance matrix for the observations from the variance parameter estimates from the reduced.asreml.obj.

extra.matrix   A matrix of order equal to the number of observations that is to be added to
                   the variance matrix, the latter based on the information in asreml.obj. It is
                   assumed that the sigma-parameterized values of the variance parameter esti-
                   mates, such as is given in the varcomp component of summary.asreml, have
                   been used in calculating extra.matrix; the values in the vparameters compo-
                   nent of G.param and R.param may be either gamma- or sigma-parameterized.
                   The argument extra.matrix can be used in conjunction with ignore.terms as
                   a workaround to include components of the variance matrix for variance func-
                   tions that have not been implemented in estimateV.

ignore.terms   A character giving terms from either the random or residual models that are
                   to be ignored in that their contributions to the variance is not to be included in the
                   estimated matrix. The term names are those given in the vparameters compo-
                   nent of the asreml object or the varcomp component produced by summary.asreml,
                   but only up to the first exclamation mark (!). This can be used in conjunction
                   with estimateV.asreml as a workaround to include components of the variance
                   matrix for variance functions that have not been implemented in estimateV.

fixed.spline.terms
                   A character vector giving one or mor spline terms in the random model that are
                   regarded as fixed and so are to be ignored because they are not regarded as con-
                   tributing to the variance. The term names are those given in the vparameters
                   component of the asreml object or the varcomp component produced by summary.asreml,
                   but only up to the first exclamation mark (!).

bound.exclusions
                   A character specifying one or more bound codes that will result in a variance
                   parameter in the random model being excluded from contributing to the vari-
                   ance. If set to NULL then none will be excluded.

tolerance   The value such that eigenvalues less than it are considered to be zero.

update   if TRUE then the arguments R.param and G.param are set to those in the asreml
                   object supplied in object so that the values from the original model are used
                   as starting values. If FALSE then calls are made to asreml in which the only
                   changes from the previous call are (i) the model is fitted to simulated data and
                   (ii) modifications specified via ... are made, except that changes cannot be
                   made to any of the models.

trace   if TRUE then partial iteration details are displayed when ASReml-R functions are
                   invoked; if FALSE then no output is displayed.

ncores   A numeric specifying the number of cores to use in doing the simulations.

...   Other arguments that are passed down to the function asreml. Changes to the
                   models are not allowed. Other changes are dangerous and generally should be
                   avoided.

Value

A list with the following components:

1. **REMLRT:** the observed REML ratio statistic.

2. **p:** the bootstrap p-value for the observed test statistic.

3. **DF:** the calculated difference in DF for the variance parameters in the two models.

4. **totalunconverged:** the total number of unconverged analyses over the simulations.

5. **REMLRT.sim:** a `numeric` containing the values of the ratio statistics for the simulated data. It has an attribute called `na.action` that can be retrieved using `attr(REMLRT.sim, which = "na.action")`; it contains a list of the simulation numbers that were abandoned because `max.retries` failed to converge for both models.

6. **nunconverged:** the number of unconverged analyses for each bootstap sample, the maximum being `max.retries`.

Note

A boostrap sample is generated using a multivariate normal distribution with expected value as specified by `means` and variance matrix given by `V`. Each simulated sample is analysed according to the reduced model and, provided this analysis converges, according to the full.model. If one of these analyses fails to converge, it is abandoned and another sample is generated for this simulation. As many as `max.retries` attempts are made to generate a data set for which both analyses converge. If data set that converges for both analyses is not generated for a simulation, NA is returned for that bootstrap sample. Hence, the maximum number of data sets that will be generated is nboot * `max.retries` and less than nboot samples will be generated if a data set that converges for both analyses is not obtained within `max.retries` attempts.

If a bootstrap sample converges for both analyses, the REML ratio test statistic is calculated as $2(log(REML)_F - log(REML)_R)$.

The `DF` is calculated from the information in `full.asreml.obj` and `reduced.asreml.obj`. The degrees of freedom are computed as the difference between the two models in the number of variance parameters whose estimates do not have a code for bound specified in `bound.exclusions`.

If ASReml-R version 4 is being used then the codes specified in `bound.exclusions` are not restricted to a subset of the default codes, but a warning is issued if a code other than these is specified. For ASReml-R version 3, only a subset of the default codes are allowed: F (Fixed), B (Boundary), C (Constrained) and S (Singular).

Author(s)

Chris Brien

See Also

[REMLRT.asreml](), [infoCriteria.asreml](), [testranfix.asrtests]()

Examples

```
## Not run:
    bootREMLRT(ICV.max, ICV.red, ncores = 2)

## End(Not run)
```

---

changeTerms.asrtests   *Adds and drops the specified sets of terms from one or both of the fixed or random model and/or replaces the residual (rcov) model with a new model.*

Description

The specified terms are simply added or dropped, without testing, from either the fixed or random model and/or the residual (rcov) model replaced. No hypothesis testing is performed, but a check is made for boundary or singular terms. A row is added to the test.summary data.frame stating which models have been changed. Convergence in fitting the model is checked and a note included in the action if there was not. All components of the asrtests object are updated.

Usage

```
## S3 method for class 'asrtests'
changeTerms(asrtests.obj,
            dropFixed = NULL, addFixed = NULL,
            dropRandom = NULL,  addRandom = NULL,
            newResidual = NULL, label = "Changed terms",
            allow.unconverged = TRUE, checkboundaryonly = FALSE,
            trace = FALSE, update = TRUE, denDF = "numeric",
            set.terms = NULL, ignore.suffices = TRUE,
            bounds = "P", initial.values = NA, ...)
```

Arguments

asrtests.obj	An asrtests object for a fitted model that is a list containing an asreml object, a wald.tab data.frame with 4 columns, and a data.frame with 5 columns that records any previous changes and tests in the fitted model.
dropFixed	A single character string in the form of a formula which, after addition of ". ~ . -" and after expansion, specifies the sum of a set of terms to be dropped from the fixed formula.
addFixed	A single character string in the form of a formula which, after addition of ". ~ . +" and expansion, specifies the sum of a set of terms to be added to the fixed formula.
dropRandom	A single character string in the form of a formula which, after addition of " ~ . -" and expansion, specifies the sum of a set of terms to be dropped from the random formula.
addRandom	A single character string in the form of a formula which, after addition of " ~ . +"and expansion, specifies the sum of a set of terms to be added to the random formula.
newResidual	A single character string in the form of a formula which, after addition of " ~ ", specifies the residual (or rcov) model. To remove the model, enter "-(.)".
label	A character string to use as the label in test.summary and which indicates what is being tested.
allow.unconverged	A logical indicating whether to accept a new model even when it does not converge. If FALSE and the fit does not converge, the supplied asreml object is returned.
checkboundaryonly	if TRUE then boundary and singular terms are not removed by rmboundary.asrtests; a warning is issued instead.
trace	If TRUE then partial iteration details are displayed when ASReml-R functions are invoked; if FALSE then no output is displayed.

update	if TRUE then update.asreml is called to fit the model to be tested. In doing this the arguments R.param and G.param are set to those in the asreml object stored in asrtests.obj so that the values from the previous model are used as starting values. If FALSE then a call is made to asreml in which the only changes from the previous call are that (i) models are modifed as specified and (ii) modifications specified via . . . are made.
denDF	Specifies the enthod to use in computing approximate denominator degrees of freedom when wald.asreml is called. Can be none to suppress the computations, numeric for numerical methods, algebraic for algebraic methods or default, the default, to autommatically choose numeric or algebraic computations depending on problem size. The denominator degrees of freedom are calculated according to Kenward and Roger (1997) for fixed terms in the dense part of the model.
set.terms	A character vector specifying the terms that are to have bounds and/or initial values set prior to fitting.
ignore.suffices	
	A logical vector specifying whether the suffices of the asreml-assigned names of the variance terms (i.e. the information to the right of an "!", other than "R!") is to be ignored in matching elements of terms. If TRUE for an element of terms, the suffices are stripped from the asreml-assigned names. If FALSE for an element of terms, the element must exactly match an asreml-assigned name for a variance term. This vector must be of length one or the same length as terms. If it is of length one then the same action is applied to the asreml-assigned suffices for all the terms in terms.
bounds	a [character](#) vector specifying the bounds to be applied to the terms specified in set.terms. This vector must be of length one or the same length as set.terms. If it is of length one then the same constraint is applied to all the terms in set.terms. If any of the bounds are equal to NA then they are left unchanged for those terms.
initial.values	A character vector specifying the initial values for the terms specified in terms. This vector must be of length one or the same length as terms. If it is of length one then the same initial value is applied to all the terms in terms. If any of the initial.values are equal to NA then they are left unchanged for those terms.
...	Further arguments passed to asreml, wald.asreml and [asrtests](#).

Value

An [asrtests](#) object, which is a list containing:

1. asreml.obj: an asreml object containing the fit of the model after all boundary and singular terms have been removed;

2. wald.tab: a 4-column data.frame containing a pseudo-anova table for the fixed terms produced by wald.asreml;

3. test.summary: a data.frame with columns term, DF, denDF, p and action. A row is added to it for each term that is dropped, added or tested or a note that several terms have been added or removed. A row contains the name of the term, the DF, the p-value and the action taken. Possible codes are: Dropped, Retained, Swapped, Unswapped, Significant, Nonsignificant, Absent, Added, Removed and Boundary. If the changed model did not converge, Unconverged will be added to the code. Note that the logical asreml.obj$converge also reflects whether there is convergence.

Author(s)

Chris Brien

See Also

asrtests, rmboundary.asrtests, testranfix.asrtests, testresidual.asrtests,
newfit.asreml, reparamSigDevn.asrtests, chooseModel.asrtests

Examples

```
## Not run:
terms <- "(Date/(Sources * (Type + Species)))"
current.asrt <- changeTerms(current.asrt, addFixed = terms)


current.asrt <- changeTerms(current.asrt, dropFixed = "A + B", denDF = "algebraic")


data(Wheat.dat)
current.asr <- asreml(yield ~ Rep + WithinColPairs + Variety,
                      random = ~ Row + Column + units,
                      residual = ~ ar1(Row):ar1(Column),
                      data=Wheat.dat)
current.asrt <- asrtests(current.asr, NULL, NULL)
current.asrt <- rmboundary(current.asrt)
# Add and drop both fixed and random terms
current.asrt <- changeTerms(current.asrt,
                            addFixed = "vRow", dropFixed = "WithinColPairs",
                            addRandom = "spl(vRow)", dropRandom = "units",
                            checkboundaryonly = TRUE)
# Replace residual with model without Row autocorrelation
current.asrt <- changeTerms(current.asrt,
                            newResidual = "Row:ar1(Column)",
                            label="Row autocorrelation")


## End(Not run)
```

---

chooseModel.asrtests	*Determines the set of significant terms taking into account the hierarchy or marginality relations and records the tests performed in a data.frame.*

Description

Performs a series of hypothesis tests taking into account the marginality of terms. In particular, a term will not be tested if it is marginal to (or nested in) one that is significant. For example, if A:B is significant, then neither A nor B will be tested. For a random term, the term is removed from the model fit, any boundary terms are removed using rmboundary.asrtests and a REML likelihood ratio test is performed using REMLRT.asreml. If it is not significant and drop.ran.ns is TRUE, the term is permanently removed from the model. Note that if boundary terms are removed, the reduced model may not be nested in the full model in which case the test is not valid. For fixed terms, the Wald tests are performed and the p-value for the term obtained. If it is not significant and drop.fix.ns is TRUE, the term is permanently removed from the model. A row is added to test.summary for each term that is tested.

Usage

```
## S3 method for class 'asrtests'
chooseModel(asrtests.obj, terms.marginality=NULL,
            alpha = 0.05, allow.unconverged = TRUE,
            checkboundaryonly = FALSE, drop.ran.ns=TRUE,
            positive.zero = FALSE, bound.test.parameters = "none",
            drop.fix.ns=FALSE, denDF = "numeric",  dDF.na = "none",
            dDF.values = NULL, trace = FALSE, update = TRUE,
            set.terms = NULL, ignore.suffices = TRUE,
            bounds = "P", initial.values = NA, ...)
```

Arguments

asrtests.obj
: an [asrtests](#) object for a fitted model that is a list containing an asreml object, a wald.tab data.frame with 4 columns, and a data.frame with 5 columns that records any previous changes and tests in the fitted model.

terms.marginality
: a square matrix of ones and zeros with row and column names being the names of the terms. The diagonal elements should be one, indicating that a term is marginal to itself. Elements should be one if the row term is marginal to the column term. All other elements should be zero.

alpha
: the significance level for the test.

allow.unconverged
: A logical indicating whether to accept a new model even when it does not converge. If FALSE and a fit when a term is removed does not converge, the term will not be removed.

checkboundaryonly
: if TRUE then boundary and singular terms are not removed by [rmboundary.asrtests](#); a warning is issued instead.

drop.ran.ns
: a logical indicating whether to drop nonsignificant random terms from the model.

positive.zero
: Indicates whether the hypothesized values for the variance components being tested are on the boundary of the parameter space. For example, this is true for positively-constrained variance components that, under the reduced model, are zero. This argument does not need to be set if bound.test.parameters is set.

bound.test.parameters
: Indicates whether for the variance components being tested, at least some of the hypothesized values are on the boundary of the parameter space. The possibilities are "none", "onlybound" and "one-and-one". The default is "none", although if it is set to "none" and positive.zero is TRUE then bound.test.parameters is taken to be "onlybound". When bound.test.parameters is set to "one-and-one", it signifies that there are two parameters being tested, one of which is bound and the other is not. For example, the latter is true for testing a covariance and a positively-constrained variance component that, under the reduced model, are zero.

drop.fix.ns
: a logical indicating whether to drop a fixed term from the model when it is nonsignificant

denDF
: Specifies the enthod to use in computing approximate denominator degrees of freedom when wald.asreml is called. Can be none to suppress the computations, numeric for numerical methods, algebraic for algebraic methods or

default, the default, to automattically choose numeric or algebraic compu-
tations depending on problem size. The denominator degrees of freedom are
calculated according to Kenward and Roger (1997) for fixed terms in the dense
part of the model.

dDF.na   the method to use to obtain substitute denominator degrees of freedom. when the
                  numeric or algebraic methods produce an NA. If dDF.na = "none", no subtitute
                  denominator degrees of freedom are employed; if dDF.na = "residual", the
                  residual degrees of freedom from asreml.obj$nedf are used; if dDF.na = "maximum",
                  the maximum of those denDF that are available, excluding that for the Intercept,
                  is used; if all denDF are NA, asreml.obj$nedf is used. If dDF.na = "supplied",
                  a vector of values for the denominator degrees of freedom is to be supplied in
                  dDF.values. Any other setting is ignored and a warning message produced.
                  Generally, substituting these degrees of freedom is anticonservative in that it is
                  likely that the degrees of freedom used will be too large.

dDF.values   A vector of values to be used when dDF.na = "supplied". Its values will be
                  used when denDF in a test for a fixed effect is NA. This vector must be the same
                  length as the number of fixed terms, including (Intercept) whose value could be
                  NA.

trace   if TRUE then partial iteration details are displayed when ASReml-R functions
                  are invoked; if FALSE then no output is displayed.

update   if TRUE then update.asreml is called in testing models. In doing this the ar-
                  guments R.param and G.param are set to those in the asreml object stored in
                  asrtests.obj so that the values from the previous model are used as starting
                  values. If FALSE then a call is made to asreml in which the only changes to
                  the asreml.obj stored in the supplied asrtests.obj are (i) to the terms in the
                  fixed and random models corresponding to terms in terms.marginality and
                  (ii) those modifications specified via ....

set.terms   a character vector specifying the terms that are to have bounds and/or initial
                  values set prior to fitting.

ignore.suffices
                  a logical vector specifying whether the suffices of the asreml-assigned names
                  of the variance terms (i.e. the information to the right of an "!", other than
                  "R!") is to be ignored in matching elements of terms. If TRUE for an element
                  of terms, the suffices are stripped from the asreml-assigned names. If FALSE
                  for an element of terms, the element must exactly match an asreml-assigned
                  name for a variance term. This vector must be of length one or the same length
                  as terms. If it is of length one then the same action is applied to the asreml-
                  assigned suffices for all the terms in terms.

bounds   a [character] vector specifying the bounds to be applied to the terms speci-
                  fied in set.terms. This vector must be of length one or the same length as
                  set.terms. If it is of length one then the same constraint is applied to all the
                  terms in set.terms. If any of the bounds are equal to NA then they are left
                  unchanged for those terms.

initial.values   a character vector specifying the initial values for the terms specified in terms.
                  This vector must be of length one or the same length as terms. If it is of length
                  one then the same initial value is applied to all the terms in terms. If any of the
                  initial.values are equal to NA then they are left unchanged for those terms.

...   further arguments passed to asreml, wald.asreml and [asrtests] via [testranfix.asrtests].

Value

A list containing:

1. asrtests.obj: an asrtests object, containing the asreml object correspondiing to the final fit, a wald.tab data.frame, and a test.summary data.frame that contains a record of the testing of the terms (see asrtests for more details);

2. sig.tests: a character vector whose elements are the the significant terms amongst those tested.

Author(s)

Chris Brien

See Also

asrtests, testranfix.asrtests, testresidual.asrtests, REMLRT.asreml, rmboundary.asrtests, newfit.asreml, changeTerms.asrtests, reparamSigDevn.asrtests

Examples

```
## Not run:
data(WaterRunoff.dat)
asreml.options(keep.order = TRUE) #required for asreml4 only
current.asr <- asreml(log.Turbidity ~ Benches + (Sources * (Type + Species)) * Date,
                      random = ~Benches:MainPlots:SubPlots:spl(xDay),
                      data = WaterRunoff.dat, keep.order = TRUE)
current.asrt <- asrtests(current.asr, NULL, NULL)
terms.treat <- c("Sources", "Type", "Species",
                 "Sources:Type", "Sources:Species")
terms <- sapply(terms.treat,
                FUN=function(term){paste("Date:",term,sep="")},
                simplify=TRUE)
terms <- c("Date", terms)
terms <- unname(terms)
marginality <-  matrix(c(1,0,0,0,0,0, 1,1,0,0,0,0,  1,0,1,0,0,0,
                         1,0,1,1,0,0, 1,1,1,0,1,0, 1,1,1,1,1,1), nrow=6)
rownames(marginality) <- terms
colnames(marginality) <- terms
choose <- chooseModel(current.asrt, marginality)
current.asrt <- choose$asrtests.obj
sig.terms <- choose$sig.terms

## End(Not run)
```

---

estimateV.asreml   *Forms the estimated variance, random or residual matrix for the ob-*
                          servations from the variance parameter estimates.

Description

Forms the estimated variance (**V**), random (**G**) or (**R**) matrix for the observations, a square symmetric matrix of order equal to the number of observations. The estimates of the variance parameters and the information about the random and residual models for which they were estimated are obtained from the asreml object. This function is not available in ASReml-R version 3.

Usage

```
## S3 method for class 'asreml'
estimateV(asreml.obj, which.matrix = "V",
          extra.matrix = NULL, ignore.terms = NULL, fixed.spline.terms = NULL,
          bound.exclusions = c("F","B","S","C"), ...)
```

Arguments

asreml.obj   An asreml object from a call to asreml in which the data argument has been
                 set.

which.matrix   A character giving the matrix that is to be formed. It must be one of "V", to
                 produce the variance matrix $\mathbf{V} = \mathbf{G} + \mathbf{R}$, "G" to produce the matrix $\mathbf{G}$, corre-
                 sponding to the random formula, or "R" to produce the matrix $\mathbf{R}$, corresponding
                 to the residual formula.

extra.matrix   A matrix of order equal to the number of observations that is to be added
                 to the matrix specified by which.matrix, the latter based on the informa-
                 tion in asreml.obj. It is assumed that the sigma-parameterized values of the
                 variance parameter estimates, such as is given in the varcomp component of
                 summary.asreml, have been used in calculating extra.matrix; the values in
                 the vparameters component of G.param and R.param may be either gamma-
                 or sigma-parameterized. The argument extra.matrix can be used in conjunc-
                 tion with ignore.terms as a workaround to include components of the variance
                 matrix for variance functions that have not been implemented in estimateV.

ignore.terms   A character giving terms from either the random or residual models that are
                 to be ignored in that their contributions to the variance is not to be included in the
                 estimated matrix. The term names are those given in the vparameters compo-
                 nent of the asreml object or the varcomp component produced by summary.asreml,
                 but only up to the first exclamation mark (!). This can be used in conjunction
                 with estimateV.asreml as a workaround to include components of the variance
                 matrix for variance functions that have not been implemented in estimateV.

fixed.spline.terms
                 A character vector giving one or mor spline terms in the random model that are
                 regarded as fixed and so are to be ignored because they are not regarded as con-
                 tributing to the variance. The term names are those given in the vparameters
                 component of the asreml object or the varcomp component produced by summary.asreml,
                 but only up to the first exclamation mark (!).

bound.exclusions
                 A character specifying one or more bound codes that will result in a variance
                 parameter in the random model being excluded from contributing to the vari-
                 ance. If set to NULL then none will be excluded.

...   Provision for passsing arguments to functions called internally - not used at
                 present.

Details

The information about the variance parameters in the fitted mixed model are obtained from the
G.param and R.param components of the asreml object. The function can deal with the following
variance functions in either the random or residual models: id, diag, us, ar1, ar2, ar3, sar, sar2,
ma1, ma2, arma, exp, gau, cor, corb and corg. All of these functions, except us, can be combined
with either v or h. It will also cope with the following functions in the random model: at, str, spl,
dev, grp, fa and rr. Additionally, it can deal with the function dsum in the residual model. For
further information see the ASReml-R User Guide Version 4 (Butler et al., 2017).

Value

A `matrix` containing the estimated variance matrix.

Author(s)

Chris Brien

References

Butler, D. G., Cullis, B. R., Gilmour, A. R., Gogel, B. J. and Thompson, R. (2017). *ASReml-R User Guide Version 4.* VSN International Ltd, <http://www.vsni.co.uk/software/asreml/>.

See Also

asreml, `simulate.asreml`, `variofaces.asreml`.

Examples

```
## Not run:
data(Wheat.dat)
current.asr <- asreml(yield ~ Rep + WithinColPairs + Variety,
                      random = ~ Row + Column + units,
                      residual = ~ ar1(Row):ar1(Column),
                      data=Wheat.dat)
# Form variance matrix based on estimated variance parameters
V <- estimateV(current.asr)


## End(Not run)
```

---

facCombine.alldiffs	*Combines several factors into one in the components of an alldiffs object*

Description

Combines several `factor`s, in the `prediction` component of `object`, into one whose `levels` are the combinations of the used `levels` of the individual `factor`s. The matching changes are made to the other components of the `alldiffs` object. The levels of the `factors` are combined using `fac.combine` from the dae package.

Usage

```
## S3 method for class 'alldiffs'
facCombine(object, factors, order="standard",
           combine.levels=TRUE, sep="_", level.length = NA, ...)
```

Arguments

`object`	An object of S3-class `alldiffs`.
`factors`	A [character](#) containing the names of [factor](#)s in the `prediction` component of `object` whose levels are to be combined.
`order`	Either `standard` or `yates`. The order in which the `levels` combinations of the [factor](#)s are to be considered as numbered when forming the `levels` of the combined [factor](#); `standard` numbers them as if they are arranged in standard order, that is with the levels of the first factor moving slowest and those of the last factor moving fastest; `yates` numbers them as if they are arranged in Yates order, that is with the levels of the first factor moving fastest and those of the last factor moving slowest.
`combine.levels`	A `logical` specifying whether the `levels` labels of the new [factor](#) are to be combined from those of the [factor](#)s being combined. The default is to use the integers from 1 to the product of the numbers of combinations of used `levels` of the individual [factor](#)s, numbering the levels according to `order`.
`sep`	A character string to separate the levels when `combine.levels = TRUE`.
`level.length`	The maximum number of characters from the the levels of factors to use in the row and column labels of the tables of pairwise differences and their p-values and standard errors.
`...`	Further arguments passed to the [factor](#) call creating the new [factor](#).

Value

A [factor](#) whose `levels` are formed form the observed combinations of the `levels` of the individual [factor](#)s.

Author(s)

Chris Brien

See Also

[as.alldiffs](#), [allDifferences.data.frame](#), [print.alldiffs](#), [sort.alldiffs](#),
[reorderClassify.alldiffs](#), fac.combine in package **dae**.

Examples

```
## Not run:
data(Smarthouse.dat)
#Set up without any sorting
m1.asr <- asreml(y1 ~ Genotype*A*B,
                 random=~Replicate/Mainplot/Subplot,
                 data=Smarthouse.dat)
current.asrt <- asrtests(m1.asr)
current.asrt <- rmboundary(current.asrt)
m1.asr <- current.asrt$asreml.obj
diffs <- predictPlus(m1.asr, classify = "Genotype:A:B",
                     wald.tab = current.asrt$wald.tab,
                     error.intervals = "Stand", tables = "none")
diffs <- facCombine(diffs, factors = c("A","B")

## End(Not run)
```

getASRemlVersionLoaded

Finds the version of asreml that is loaded and returns the initial characters in version.

Description

Checks that asreml is loaded and, if it is, returns the first nchar characters of the version that is loaded.

Usage

```
getASRemlVersionLoaded(nchar = NULL, notloaded.fault = FALSE)
```

Arguments

nchar   The number of characters in the asreml version to get.

notloaded.fault

A [logical](#) indicating whether a fault is to occur if asreml is not loaded.

Value

A character, being the first nchar characters of the version of asreml that is loaded.

Author(s)

Chris Brien

See Also

[loadASRemlVersion](#).

Examples

```
## Not run:
getASRemlVersionLoaded()
## End(Not run)
```

infoCriteria.asreml   *Computes AIC and BIC for a model.*

Description

Computes Akiake and Bayesian (Schwarz) Information Criteria for a model.

Usage

```
## S3 method for class 'asreml'
infoCriteria(asreml.obj, DF = NULL,
             bound.exclusions = c("F","B","S","C"), ...)
```

Arguments

asreml.obj   An asreml object resulting from the fitting of a model using REML.

DF   A numeric giving the number of estimated variance parameters. If NULL then
                  this is determined from the information in asreml.obj.

bound.exclusions

                  A character specifying the bound (constraint) codes that will result in a vari-
                  ance parameter being excluded from the count of estimated variance parameters.
                  If set to NULL then none will be excluded.

...   Provision for passsing arguments to functions called internally - not used at
                  present.

Details

The degrees of freedom (DF) are the number of number of variance parameters that have been
estimated, excluding those whose estimates have a code for bound specified in bound.exclusions.
If DF is not NULL, the supplied value is used. Otherwise DF is determined from the information in
asreml.obj.

If ASReml-R version 4 is being used then the codes specified in bound.exclusions are not re-
stricted to a subset of the default codes, but a warning is issued if a code other than these is specified.
For ASReml-R version 3, only a subset of the default codes are allowed: F (Fixed), B (Boundary),
C (Constrained) and S (Singular).

The AIC is calculated as $-2 \times log(REML) + 2 \times DF$ and the BIC as $-2 \times log(REML) + DF \times (n - p)$, where n is the number of observations and r is the rank of the fixed effects design matrix.

Value

A data frame containing the degrees of freedom, number of bound parameters, AIC, BIC and log of
the REML value.

Author(s)

Chris Brien

See Also

REMLRT.asreml

Examples

```
## Not run:
  data(Wheat.dat)
  current.asr <- asreml(yield ~ Rep + WithinColPairs + Variety,
                        random = ~ Row + Column + units,
                        residual = ~ ar1(Row):ar1(Column),
                        data=Wheat.dat)
   infoCriteria(current.asr)

## End(Not run)
```

linTransform.alldiffs   *Calculates a linear transformation of the predictions stored in an alldiffs object.*

Description

Effects the linear transformation of the predictions in the supplied alldiffs object, the transformation being specified by a matrix or a formula. The values of the transformed values are stored in an alldiffs object. A matrix might be a contrast matrix or a matrix of weights for the levels of a factor used to obtain the weighted average over the levels of that factor. A formula gives rise to a projection matrix that linearly transforms the predictions so that they conform to the model specified by the formula, this model being a submodel of that inherent in the classify.

If pairwise = TRUE, all pairwise differences between the linear transforms of the predictions, their standard errors, p-values and LSD statistics are computed as using allDifferences.data.frame. This adds them to the alldiffs object as additional list components named differences, sed, p.differences and LSD.

If a transformation has been applied (any one of transform.power is not one, scale is not one and offset is nonzero), the backtransforms of the transformed values and their lower and upper confidence intervals are added to a data.frame that is consistent with an object of class asremlPredict, such as is stored in the pvals component of the prediction component of the value produced by predict.asreml. If transform.power is other than one, the standard.error column of the data.frame is set to NA. This data.frame is added to the alldiffs object as a list component called backtransforms.

The printing of the components produced is controlled by the tables argument. The order of plotting the levels of one of the factors indexing the predictions can be modified and is achieved using sort.alldiffs.

Usage

```
## S3 method for class 'alldiffs'
linTransform(alldiffs.obj, classify = NULL, term = NULL,
             linear.transformation = NULL, Vmatrix = FALSE,
             error.intervals = "Confidence", avsed.tolerance = 0.25,
             meanLSD.type = "overall", LSDby = NULL,
             response = NULL, response.title = NULL,
             x.num = NULL, x.fac = NULL,
             tables = "all", level.length = NA,
             pairwise = TRUE, alpha = 0.05,
             transform.power = 1, offset = 0, scale = 1,
             inestimable.rm = TRUE, ...)
```

Arguments

alldiffs.obj   An object of S3-class alldiffs (see as.alldiffs).

classify   a character string giving the variables that define the margins of the multiway table to be predicted. Multiway tables are specified by forming an interaction type term from the classifying variables, that is, separating the variable names with the : operator.

term   a [character](#) string giving the variables that define the term that was fitted using
                        asreml and that corresponds to classify. It only needs to be specified when it
                        is different to classify.

linear.transformation

                        a [formula](#) or a [matrix](#). If a [formula](#) is given then it is taken to be a submodel
                        of the model term corresponding to the classify. The projection matrix that
                        transforms the predictions so that they conform to the submodel is obtained;
                        the submodel should involving the variables in the classify. For example, for
                        classify set to "A:B", the submodel ~ A + B will result in the predictions
                        for the combinations of A and B being made additive for the [factor](#)s A and B.

                        If a [matrix](#) is provided then it will be used to apply the linear transformation to
                        the predictions. The number of rows in the [matrix](#) should equal the number
                        of linear combinations of the predictions desired and the number of columns
                        should equal the number of predictions.

                        In either case, as well as the values of the linear combinations, their standard
                        errors, pairwise differences and associated statistics are returned.

Vmatrix   A [logical](#) indicating whether the variance matrix of the predictions will be
                        stored as a component of the alldiffs object that is returned. If linear.transformation
                        is set, it will be stored irrespective of the value of Vmatrix.

error.intervals

                        A [character](#) string indicating the type of error interval, if any, to calculate
                        in order to indicate uncertainty in the results. Possible values are "none",
                        "StandardError", "Confidence" and "halfLeastSignificant". The de-
                        fault is for confidence limits to be used. The "halfLeastSignificant" option
                        results in half the Least Significant Difference (LSD) being added and subtracted
                        to the predictions, the LSD being calculated using the square root of the mean of
                        the variances of all or a subset of pairwise differences between the predictions. If
                        meanLSD.type is set to overall, the avsed.tolerance is not NA and the range
                        of the SEDs divided by the average of the SEDs exceeds avsed.tolerance
                        then the error.intervals calculations and the plotting will revert to confi-
                        dence intervals. Also, half LSDs cannot be used for backtansformed values and
                        so confidence intervals will be used instead.

avsed.tolerance

                        A [numeric](#) giving the value of the SED range, the range of the SEDs divided
                        by the square root of the mean of the variances of all or a subset of the pairwise
                        differences, that is considered reasonable in calculating error.intervals. It
                        should be a value between 0 and 1. The following rules apply:

                        1. If avsed.tolerance is NA then mean LSDs of the type specified by meanLSD.type
                           are calculated and used in error.intervals and plots.
                        2. Irrespective of the setting of meanLSD.type, if avsed.tolerance is not
                           exceeded then the mean LSDs are used in error.intervals and plots.
                        3. If meanLSD.type is set to overall, avsed.tolerance is not NA, and avsed.tolerance
                           is exceeded then error.intervals and plotting revert to confidence inter-
                           vals.
                        4. If meanLSD.type is set to factor.combinations and avsed.tolerance
                           is not exceeded for any factor combination then the half LSDs are used
                           in error.intervals and plots; otherwise, error.intervals and plotting
                           revert to confidence intervals.
                        5. If meanLSD.type is set to per.prediction and avsed.tolerance is not
                           exceeded for any prediction then the half LSDs are used in error.intervals

and plots; otherwise, `error.intervals` and plotting revert to confidence intervals.

meanLSD.type   A [character](#) string determining whether the mean LSD stored is (i) the `overall` mean, based on the square root of the mean of the variances of all pairwise variances, (ii) the mean for each `factor.combination` of the [factor](#)s specified by LSDby, which is based on the square root of the mean of the variances for all pairwise differences for each factor combination, or (iii) the `per.prediction` mean, based, for each prediction, on the square root of the mean of the variances for all pairwise differences involving that prediction. It also determines, in conjunction with `avsed.tolerance`, which LSD will be used in calculating `error.intervals` and, hence, is used for plots.

LSDby   A [character](#) (vector) of factor names, being the names of the [factor](#)s for each of whose levels combinations a mean LSD, minLSD and max LSD is stored in the LSD component of the `alldiffs` object when `meanLSD.type` is `factor.combinatons`.

response   a character specifying the response variable for the predictions. It is stored as an attribute to the `alldiffs` object.

response.title a character specifying the title for the response variable for the predictions. It is stored as an attribute to the `alldiffs` object.

x.num   A [character](#) string giving the name of the numeric covariate that (i) corresponds to `x.fac`, (ii) is potentially included in terms in the fitted model, and (iii) which corresponds to the x-axis variable. It should have the same number of unique values as the number of levels in `x.fac`.

x.fac   A [character](#) string giving the name of the factor that (i) corresponds to `x.num`, (ii) is potentially included in terms in the fitted model, and (iii) which corresponds to the x-axis variable. It should have the same number of levels as the number of unique values in `x.num`. The levels of `x.fac` must be in the order in which they are to be plotted - if they are dates, then they should be in the form yyyymmdd, which can be achieved using `as.Date`. However, the levels can be non-numeric in nature, provided that `x.num` is also set.

tables   A [character](#) vector containing a combination of none, predictions, vcov, backtransforms, differences, p.differences, sed, LSD and all. These nominate which components of the `alldiffs` object to print.

level.length   The maximum number of characters from the the levels of factors to use in the row and column labels of the tables of pairwise differences and their p-values and standard errors.

pairwise   A [logical](#) indicating whether all pairwise differences of the predictions and their standard errors and p-values are to be computed and stored. If tables is equal to "differences" or "all" or error.intervals is equal to "halfLeastSignificant", they will be stored irrespective of the value of pairwise.

alpha   A [numeric](#) giving the significance level for LSDs or one minus the confidence level for confidence intervals.

transform.power

A [numeric](#) specifying the power of a transformation, if one has been applied to the response variable. Unless it is equal to 1, the default, back-transforms of the predictions will be obtained and presented in tables or graphs as appropriate. The back-transformation raises the predictions to the power equal to the reciprocal of `transform.power`, unless it equals 0 in which case the exponential of the predictions is taken.

offset   A [numeric](#) that has been added to each value of the response after any scaling
                and before applying any power transformation.

scale   A [numeric](#) by which each value of the response has been multiplied before
                adding any offset and applying any power transformation.

inestimable.rm A logical indicating whether rows for predictions that are not estimable are to
                be removed from the components of the alldiffs object.

...   provision for passsing arguments to functions called internally - not used at
                present.

Details

For a matrix $\mathbf{L}$, vector of predictions $\mathbf{p}$ and variance matrix of the predictions $\mathbf{V}_p$, the linear trans-
formed predictions are given by $\mathbf{Lp}$ with variance matrix $\mathbf{LV}_p\mathbf{L}^{\mathrm{T}}$. The last matrix is used to com-
pute the variance of pairwise differences between the transformed values.

The [matrix](#) $\mathbf{L}$ is directly specified by setting linear.transformation to it. If linear.transformation
is a [formula](#) then $\mathbf{L}$ is formed as the sum of the orthogonal projection matrices obtained using
pstructure.formula from the package dae; grandMean is set to TRUE and orthogonalize to
"eigenmethods".

Value

A alldiffs object with the linear transformation of the predictions and their standard errors and
all pairwise differences between the linear transforms of their predictions, their standard errors and
p-values and LSD statistics.

If transform.power or scale is not one or offset is not zero, it will contain a data.frame with
the backtransformed linear transformation of the predictions. The backtransformation will, after
backtransforming for any power transformation, subtract the offset and then divide by the scale.

If error.intervals is not "none", then the predictions component and, if present, the backtransforms
component will contain columns for the lower and upper values of the limits for the interval. The
names of these columns will consist of three parts separated by full stops: 1) the first part will be
lower or upper; 2) the second part will be one of Confidence, StandardError or halfLeastSignificant;
3) the third component will be limits.

The name of the response, the response.title, the term, the classify, tdf, sortFactor and
the sortOrder will be set as attributes to the object.

Author(s)

Chris Brien

See Also

[predictPlus.asreml](#), [as.alldiffs](#), [print.alldiffs](#), [sort.alldiffs](#), [subset.alldiffs](#),
[allDifferences.data.frame](#), [redoErrorIntervals.alldiffs](#), [recalcLSD.alldiffs](#),
[predictPresent.asreml](#), [plotPredictions.data.frame](#), as.Date, predict.asreml

Examples

```
## Not run:
data(WaterRunoff.dat)
asreml.options(keep.order = TRUE) #required for asreml4 only
current.asr <- asreml(fixed = pH ~ Benches + (Sources * (Type + Species)),
                      random = ~ Benches:MainPlots,
```

```
                              keep.order=TRUE, data= WaterRunoff.dat)
current.asrt <- asrtests(current.asr, NULL, NULL)
#Get additive predictions directly using predictPlus
diffs.sub <- predictPlus.asreml(classify = "Sources:Species", Vmatrix = TRUE,
                                linear.transformation = ~ Sources + Species,
                                asreml.obj = current.asr, tables = "none",
                                wald.tab = current.asrt$wald.tab,
                                present = c("Type","Species","Sources"))
#Contrast matrix for differences between each species and non-planted for the last source
L <- cbind(matrix(rep(0,7*32), nrow = 7, ncol = 32),
           diag(1, nrow = 7),
           matrix(rep(-1, 7), ncol = 1))
rownames(L) <- as.character(diffs.sub$predictions$Species[33:39])
diffs.L <- linTransform(diffs.sub,
                        classify = "Sources:Species",
                        linear.transformation = L,
                        tables = "predictions")

## End(Not run)
```

---

loadASRemlVersion   *Ensures that a specific version of* asreml *is loaded.*

Description

Loads the specified version of asreml, provided that it is not already loaded. If the version of asreml is not the required version, then the loaded version is unloaded first.

Usage

```
loadASRemlVersion(version = 4, ...)
```

Arguments

version   The version that is to be loaded, the vesion consisting of just the initial char-
             acters that are significant in the version that should be loaded. For example, the
             default value of 4 implies that any version that begins with "4" is acceptable. It
             is used to check that the required version is loaded.

...   Other library/require arguments that are needed to load the specified version of
             asreml.

Value

A character, being all characters in the version of asreml that is loaded on exit from the function.

Author(s)

Chris Brien

See Also

[getASRemlVersionLoaded](#).

Examples

```
## Not run:
loadASRemlVersion(3, lib.loc = "D:\Analyses\R asreml3")
## End(Not run)
```

---

newfit.asreml	*Refits an asreml model with modified model formula using either a call to* update.asreml *or a direct call to* asreml.

Description

Extracts the call from the asreml.obj and evaluates that call, replacing any arguments with changed values. If update is TRUE and set.terms is not set, the call is evaluated using update.asreml; otherwise, it is evaluated using a direct call to asreml. The principal difference is that the latter does not enforce the use of previous values of the variance parameters as initial values; it sets G.param and R.param to NULL or to values as specified for set.terms. The ... argument can be used to pass G.param and/or R.param, provided update is FALSE and set.terms is not set.

Usage

```
## S3 method for class 'asreml'
newfit(asreml.obj, fixed., random., sparse.,
       residual., rcov., update = TRUE,
       allow.unconverged = TRUE, keep.order = TRUE,
       set.terms = NULL, ignore.suffices = TRUE, bounds = "P",
       initial.values = NA, ...)
```

Arguments

asreml.obj
: a valid asreml object with with a component named call (from a previous call to either asreml or update.asreml).

fixed.
: a character or formula specifying changes to the fixed formula. This is a two-sided formula where "." is substituted for existing components in the fixed component of asreml.obj$call.

random.
: a character or formula specifying changes to the random formula. This is a one-sided formula where "." is substituted for existing components in the random component of asreml.obj$call.

sparse.
: a character or formula specifying changes to the sparse formula. This is a one-sided formula where "." is substituted for existing components in the sparse component of asreml.obj$call.

residual.
: a character or formula specifying changes to the error formula, used when version 4 or later of ASReml-R is loaded. This is a one-sided formula where "." is substituted for existing components in the residual component of asreml.obj$call.

rcov.
: a character or formula specifying changes to the error formula, used when version 3 of ASReml-R is loaded. This is a one-sided formula where "." is substituted for existing components in the residual component of asreml.obj$call.

update	a logical indicated whether to use `update.asreml` or `asreml` to evaluate the modified call. If `TRUE`, use `update.asreml` to evaluate the modified `call`. In doing this the arguments `R.param` and `G.param` are set to those in the `asreml.obj` so that the values from the previous model are used as starting values. If `FALSE` then a call is made to `asreml` itself, in which the only changes from the previous call are those specified in the arguments to `newfit.asreml`.
allow.unconverged	
	A `logical` indicating whether to accept a new model even when it does not converge. If `FALSE` and the fit does not converge, the supplied `asreml` object is returned.
keep.order	a logical value indicating whether the terms should keep their positions. If `FALSE` the terms are reordered so that main effects come first, followed by the interactions, all second-order, all third-order and so on. Effects of a given order are kept in the order specified.
set.terms	a character vector specifying the terms that are to have bounds and/or initial values set prior to fitting.
ignore.suffices	
	a logical vector specifying whether the suffices of the `asreml`-assigned names of the variance terms (i.e. the information to the right of an "!", other than "R!") is to be ignored in matching elements of `terms`. If `TRUE` for an element of `terms`, the suffices are stripped from the `asreml`-assigned names. If `FALSE` for an element of `terms`, the element must exactly match an `asreml`-assigned name for a variance term. This vector must be of length one or the same length as `terms`. If it is of length one then the same action is applied to the `asreml`-assigned suffices for all the terms in `terms`.
bounds	a [character](#) vector specifying the bounds to be applied to the terms specified in `set.terms`. This vector must be of length one or the same length as `set.terms`. If it is of length one then the same constraint is applied to all the terms in `set.terms`. If any of the bounds are equal to NA then they are left unchanged for those terms.
initial.values	a character vector specifying the initial values for the terms specified in `terms`. This vector must be of length one or the same length as `terms`. If it is of length one then the same initial value is applied to all the terms in `terms`. If any of the initial.values are equal to NA then they are left unchanged for those terms.
...	additional arguments to the call, or arguments with changed values.

Value

An `asreml` object.

Author(s)

Chris Brien

References

Butler, D. G., Cullis, B. R., Gilmour, A. R., Gogel, B. J. and Thompson, R. (2017). *ASReml-R User Guide Version 4*. VSN International Ltd, <http://www.vsni.co.uk/software/asreml/>.

See Also

update.asreml, [setvarianceterms.call](#)

Examples

```
## Not run:
    m2.asreml <- newfit(m1.asreml, random. = "~ . - Blocks:Plots", maxiter=75)

## End(Not run)
```

---

num.recode   *Recodes the unique values of a vector using the values in a new vector.*

Description

Recodes the unique values of a variate using the value in position i of the new.values vector to replace the ith sorted unique values of x. The new levels do not have to be unique.

Usage

```
num.recode(x, new.values)
```

Arguments

x	The vector to be recoded.
new.values	A vector of length unique(x) containing values to use in the recoding.

Value

A vector.

Author(s)

Chris Brien

See Also

dae::fac.recode.

Examples

```
## set up a factor with labels
x <- rep(c(-42, -14, 14, 42), 4)

## recode x
b <- num.recode(x, c(0, 28, 56, 84))
```

Oats.dat	*Data for an experiment to investigate nitrogen response of 3 oats varieties*

Description

Yates (1937) describes a split-plot experiment that investigates the effects of three varieties of oats and four levels of Nitrogen fertilizer. The varieties are assigned to the main plots using a randomized complete block design with 6 blocks and the nitrogen levels are randomly assigned to the subplots in each main plot.

The columns in the data frame are: Blocks, Wplots, Subplots, Variety, Nitrogen, xNitrogen, Yield. The column xNitrogen is a numeric version of the factor Nitrogen. The response variable is Yield.

Usage

```
data(Oats.dat)
```

Format

A data.frame containing 72 observations of 7 variables.

Author(s)

Chris Brien

Source

Yates, F. (1937). The Design and Analysis of Factorial Experiments. *Imperial Bureau of Soil Science, Technical Communication*, **35**, 1-95.

---

permute.square	*Permutes the rows and columns of a square matrix.*

Description

Permutes the rows and columns of a square matrix.

Usage

```
permute.square(x, permutation)
```

Arguments

x	A square matrix.
permutation	A vector specifying the new order of rows and columns.

Value

A square `matrix`.

Author(s)

Chris Brien

See Also

permute.to.zero.lowertri

Examples

```
terms.marginality <-  matrix(c(1,0,0,0,0,  0,1,0,0,0, 0,1,1,0,0,
                               1,1,1,1,0, 1,1,1,1,1), nrow=5)
permtn <- c(1,3,2,4,5)
terms.marginality <- permute.square(terms.marginality, permtn)
```

---

permute.to.zero.lowertri

Permutes a square matrix until all the lower triangular elements are zero.

Description

Permutes a square matrix until all the lower triangular elements are zero.

Usage

```
permute.to.zero.lowertri(x)
```

Arguments

x   A square matrix of order n with at least n*(n-1)/2 zero elements.

Value

A square matrix.

Author(s)

Chris Brien

See Also

permute.square

Examples

```
terms.marginality <-  matrix(c(1,0,0,0,0,  0,1,0,0,0, 0,1,1,0,0,
                               1,1,1,1,0, 1,1,1,1,1), nrow=5)
terms.marginality <- permute.to.zero.lowertri(terms.marginality)
```

plotPredictions.data.frame
Plots the predictions for a term, possibly with error bars.

Description

This function plots the predictions y that are based on `classify` and stored in the `data.frame` `data`. The package `ggplot2` is used to produce the plots. Line plots are produced when variables involving `x.num` or `x.fac` are involved in `classify` for the predictions; otherwise, bar charts are produced. Further, for line charts, the argument `panels` determines whether a single plot or multiple plots in a single window are produced; for bar charts, the argument `panels` is ignored.

Usage

```
## S3 method for class 'data.frame'
plotPredictions(data, classify, y,
                x.num = NULL, x.fac = NULL, nonx.fac.order = NULL,
                colour.scheme = "colour", panels = "multiple",
                graphics.device = NULL,
                error.intervals = "Confidence",  titles = NULL,
                y.title = NULL, filestem = NULL, ggplotFuncs = NULL, ...)
```

Arguments

data	a `data.frame` containing the values of the variables to be plotted. It should be consistent with an object of class `asremlPredict` or `asreml.predict` such as is stored in the `pvals` component of the value produced by `predict.asreml`; that is, in addition to variables classifying the predictions, it will include a column with the name specified in the y argument, usually `predicted.value` or `backtransformed.predictions`; each row contains a single predicted value. If `error.intervals` is not `"none"`, then the `predictions` component and, if present, the `backtransforms` component should contain columns for the lower and upper values of the limits for the interval with names that begin with `lower` and `upper`, respectively. The second part of the name must be one of `Confidence`, `StandardError` or `halfLeastSignificant`. The last part needs to be consistent between the lower and upper limits.
classify	a character string giving the combinations of the independent variables on which the predictions are based. It is an interaction type term formed from the independent variables, that is, separating the variable names with the : operator.
y	a character string giving the name of the variable that is to be plotted on the Y axis.
x.num	A character string giving the name of the numeric covariate that corresponds to `x.fac`, is potentially included in terms in the fitted model and which corresponds to the x-axis variable. It should have the same number of unique values as the number of levels in `x.fac`.
x.fac	A character string giving the name of the factor that corresponds to `x.num`, is potentially included in terms in the fitted model and which corresponds to the x-axis variable. It should have the same number of levels as the number of unique values in `x.num`. The levels of `x.fac` must be in the order in which they are to be plotted - if they are dates, then they should be in the form yyyymmdd, which can

	be achieved using `as.Date`. However, the levels can be non-numeric in nature, provided that `x.num` is also set.
nonx.fac.order	A character vector giving the order in which factors other than `x.fac` are to be plotted in facetted plots (i.e. where the number of non x factors is greater than 1). The first factor in the vector will be plotted on the X axis (if there is no `x.num` or `x.fac`. Otherwise, the order of plotting the factors is in columns (X facets) and then rows (Y facets). By default the order is in decreasing order for the numbers of levels of the non x factors.
colour.scheme	A character string specifying the colour scheme for the plots. The default is `"colour"` which produces coloured lines and bars, a grey background and white gridlines. A value of `"black"` results in black lines, grey bars and gridlines and a white background.
panels	Possible values are `"single"` and `"multiple"`. When line plots are to be produced, because variables involving `x.num` or `x.fac` are involved in `classify` for the predictions, `panels` determines whether or not a single panel or multiple panels in a single window are produced. The `panels` argument is ignored for bar charts.
graphics.device	A `character` specifying a graphics device for plotting. The default is `graphics.device = NULL`, which will result in plots being produced on the current graphics device. Setting it to `"windows"`, for example, will result in a windows graphics device being opened.
error.intervals	A character string indicating the type of error interval, if any, to plot in order to indicate uncertainty in the results. Possible values are `"none"`, `"StandardError"`, `"Confidence"` and `"halfLeastSignificant"`. Here, any option other than `"none"` will result in the interval limits contained in `data` being plotted.
titles	A `list`, each component of which is named for a column in the `data.frame` for the `asreml.obj` used in making the predictions and contains a `character string` giving a title to use in output (e.g. tables and graphs). Here they will be used for axis labels for nonresponse variables. For response variable labels see `y.title`.
filestem	A character sting giving the beginning of the name of the file in which to save the plot. If `filestem = NULL`, the plot is not saved. The remainder of the file name will be generated automatically and consists of the following elements separated by full stops: the classify term, `Bar` or `Line` and, if `error.intervals` is not `"none"`, one of `SE`, `CI` or `LSI`. The file will be saved as a 'png' file in the current work directory.
y.title	The title to be displayed on the y axis of any plot.
ggplotFuncs	A [list](#), each element of which contains the results of evaluating a [ggplot](#) function. It is created by calling the [list](#) function with a [ggplot](#) function call for each element.
...	further arguments passed to ggplot.

Value

no values are returned.

Author(s)

Chris Brien

See Also

allDifferences.data.frame, predictPresent.asreml, redoErrorIntervals.alldiffs, recalcLSD.alldiffs, ggplot, Devices

Examples

```
## Not run:
data(WaterRunoff.dat)
asreml.options(keep.order = TRUE) #required for asreml4 only
current.asr <- asreml(fixed = log.Turbidity ~ Benches + Sources + Type + Species +
                                 Sources:Type + Sources:Species +
                                 Sources:xDay + Species:xDay + Species:Date,
                      data = WaterRunoff.dat, keep.order = TRUE)
current.asrt <- asrtests(current.asr, NULL, NULL)

#### Get the observed combinations of the factors and variables in classify
class.facs <- c("Species","Date","xDay")
levs <- as.data.frame(table(WaterRunoff.dat[class.facs]))
levs <- as.list(levs[levs$Freq != 0, class.facs])
levs$xDay <- as.numfac(levs$xDay)

predictions <- predict(current.asr, classify="Species:Date:xDay",
                        parallel = TRUE, levels = levs,
                        present = c("Type","Species","Sources"))
#### for asreml3
predictions <- predictions$predictions$pvals
predictions <- predictions[predictions$est.status == "Estimable",]
#### for asreml4
predictions <- predictions$pvals
predictions <- predictions[predictions$status == "Estimable",]

plotPredictions(classify="Species:Date:xDay", y = "predicted.value",
                data = predictions, wald.tab = current.asrt$wald.tab,
                x.num = "xDay", x.fac = "Date",
                x.title = "Days since first observation",
                y.title = "Predicted log(Turbidity)",
                present = c("Type","Species","Sources"),
                error.intervals = "none",
                ggplotFuncs = list(ggtitle("Transformed turbidity over time")))

diffs <- predictPlus(classify="Species:Date:xDay",
                     present=c("Type","Species","Sources"),
                     asreml.obj = current.asr, tables = "none",
                     x.num = "xDay", x.fac = "Date",
                     parallel = TRUE, levels = levs,
                     x.plot.values=c(0,28,56,84),
                     wald.tab = current.asrt$wald.tab)
x.title <- "Days since first observation"
names(x.title) <- "xDay"
plotPredictions(classify="Species:Date:xDay", y = "predicted.value",
                data = diffs$predictions, wald.tab = current.asrt$wald.tab,
                x.num = "xDay", x.fac = "Date",
                titles = x.title,
                y.title = "Predicted log(Turbidity)")

## End(Not run)
```

---

plotPvalues.alldiffs   *Plots a heat map of p-values for pairwise differences between predictions.*

Description

Produces a heat-map plot of the p-values for pairwise differences between predictions that is stored in the `p.differences` component of an `all.diffs` object. This is generally a matrix whose rows and columns are labelled by the levels of one or more factors, the set of labels being the same for rows and columns. The `sections` argument allows multiple plots to be produced, one for each combination of the levels of the factors listed in `sections`. Otherwise, a single plot is produced for all observed combinations of the levels of the factors in the `classify` for the `alldiffs` object. The plots are produced using `plotPvalues.data.frame`. The order of plotting the levels of one of the factors indexing the predictions can be modified and is achieved using `sort.alldiffs`.

Usage

```
plotPvalues(object, ...)
## S3 method for class 'alldiffs'
plotPvalues(object, sections = NULL,
            gridspacing = 0, factors.per.grid = 0,
            show.sig = FALSE, triangles = "both",
            title = NULL, axis.labels = TRUE, sep=",",
            colours = RColorBrewer::brewer.pal(3, "Set2"),
            ggplotFuncs = NULL, sortFactor = NULL,
            sortWithinVals = NULL, sortOrder = NULL,
            decreasing = FALSE, ...)
```

Arguments

object   An `alldiffs` object with a `p.differences` component that is not NULL. with either 3 or 4 columns. Only if there are 4 columns, the first should be a factor indexing sections for which separate variogram plots are to be produced. In either case, the other 3 columns should be, in order, (i) a factor indexing the x-direction, (ii) a factor indexing the y-direction, and (iii) the residuals for the observed response.

sections   A `character` listing the names of the factors that are to be used to break the plot into sections. A separate plot will be produced for each observed combination of the levels of these factors.

gridspacing   A `numeric` specifying the number(s) of rows and columns that form groups in the grid of differences. An alternative is to specify the `factors.per.grid` argument to have the grid spacings automatically calculated. Grids are most useful when two or more factors index the rows and columns. If a single, nonzero number, *k* say, is given then a grid line is placed after every *k*th row and column. If a vector of values is given then the number of grid lines is the length of the vector and the spacing between each is specified by the elements of the vector.

factors.per.grid

                     A `numeric` specifying the number of factors to include within each grid of differences. The `gridspacing` will then be computed based on the numbers of combinations observed within the levels of the remaining factors in a single plot. The

	gridspacing argument to this function will be ignored if `factors.per.grid` is greater than zero. Grids are most useful when two or more factors index the rows and columns of each plot.
show.sig	A `logical` that specifies whether asterisks indicating the level of significance are to be added to the plot. If they are then '***' indicates that $p \leq 0.001$, '**' that $0.001 < p \leq 0.01$, '*' that $0.01 < p \leq 0.05$ '.' that $0.05 < p \leq 0.10$.
triangles	A `character` indicating whether the plot should include the `lower`, `upper` or `both` traingle(s).
title	A `character` string giving the main title for the plot and to which is appended the levels combination of the sectioning factors, if any, for each plot.
axis.labels	A `logical` indicating whether a label is to be added to the x- and y-axes. If TRUE, the label is the comma-separated list of factors whose levels combinations are involved in the prediction differences for which the p-values are calculated.
colours	A vector of of colours to be passed to the ggplot function `scale_colour_gradientn`.
sep	A `character` giving the characters separating the levels of different factors in the row and column names of the `p.differences` component.
ggplotFuncs	A [list](#), each element of which contains the results of evaluating a [ggplot](#) function. It is created by calling the [list](#) function with a [ggplot](#) function call for each element. It is passed to ggplot via [plotPvalues.data.frame](#).
sortFactor	A `character` containing the name of the `factor` that indexes the set of predicted values that determines the sorting of the `alldiffs` components by [sort.alldiffs](#). If NULL then sorting is not carried out. If there is more than one variable in the `classify` term then `sortFactor` is sorted for the predicted values within each combination of the values of the sortWithin variables: the `classify` variables, excluding the `sortFactor`. There should be only one predicted value for each unique value of `sortFactor` within each set defined by a combination of the values of the sortWithin variables.
sortWithinVals	A list with a component named for each `factor` and `numeric` that is a `classify` variable for the predictions, excluding `sortFactor`. Each component should contain a single value that is a value of the variable. The combination of this set of values will be used to define a subset of the predicted values whose order will define the order of `sortFactor` to be used for all combinations of the `sortWithinVals` variables. If `sortWithinVals` is NULL then the first value of each sortWithin variable in `predictions` component is used to define `sortWithinVals`. If there is only one variable in the classify then `sortWithinVals` is ignored.
sortOrder	A `character` vector whose length is the same as the number of levels for `sortFactor` in the `predictions` component of the `alldiffs` object. It specifies the desired order of the levels in the reordered components of the `alldiffs` object. The argument `sortWithinVals` is ignored.
	The following creates a `sortOrder` vector `levs` for factor `f` based on the values in x: `levs <- levels(f)[order(x)]`.
decreasing	A `logical` passed to `order` that detemines whether the order for sorting the `alldiffs` components is for increasing or decreasing magnitude of the predicted values.
...	Provision for passssing arguments to functions called internally - not used at present.

Value

A `data.frame` with the columns X1, X2, p, sections1 and sections2. This data.frame is formed from the `p.differences` component of `object` and is used in producng the plot.

Author(s)

Chris Brien

See Also

[plotPvalues.data.frame](), [allDifferences.data.frame](), [sort.alldiffs](), [subset.alldiffs](),
ggplot

Examples

```
## Not run:
data(WaterRunoff.dat)
asreml.options(keep.order = TRUE) #required for asreml4 only
current.asr <- asreml(fixed = pH ~ Benches + (Sources * (Type + Species)),
                      random = ~ Benches:MainPlots,
                      keep.order=TRUE, data= WaterRunoff.dat))
current.asrt <- asrtests(current.asr, NULL, NULL)
diffs <- predictPlus.asreml(classify = "Sources:Type",
                            asreml.obj = current.asr, tables = "none",
                            wald.tab = current.asrt$wald.tab,
                            present = c("Type","Species","Sources"))

plotPvalues(diffs, gridspacing = rep(c(3,4), c(4,2)), show.sig = TRUE)
plotPvalues(diffs, sections = "Sources", show.sig = TRUE, axis.labels = TRUE)

## End(Not run)
```

---

plotPvalues.data.frame

Plots a heat map of p-values for pairwise differences between predic-
tions.

Description

Produces a heat-map plot of the p-values for pairwise differences between predictions that is in a
`data.frame`. The `data.frame` includes two factors whose levels specify, for each p-value, which
factor levels are being compared.

Usage

```
## S3 method for class 'data.frame'
plotPvalues(object, p = "p", x, y,
            gridspacing = 0, show.sig = FALSE, triangles = "both",
            title = NULL, axis.labels = NULL,
            colours = RColorBrewer::brewer.pal(3, "Set2"),
            ggplotFuncs = NULL, ...)
```

Arguments

object	A data.frame containing the three columns specified by p, x and y.
p	A character giving the name of the column in object that contains the p-values to be plotted.
x	A character giving the name of the column in object that contains the factor whose levels index the p-values that are to be plotted in the same column.
y	A character giving the name of the column in object that contains the labels of the the p-values that are to be plotted as the rows.
gridspacing	A numeric specifying the number(s) of rows and columns that form groups in the grid of differences. This is most useful when two or more factors index the rows and columns. If a single, nonzero number, *k* say, is given then a grid line is placed after every *k*th row and column. If a vector of values is given then the number of grid lines is the length of the vector and the spacing between each is specified by the elements of the vector.
show.sig	A logical that specifies whether asterisks indicating the level of significance are to be added to the plot. If they are then '***' indicates that $p \leq 0.001$, '**' that $0.001 < p \leq 0.01$, '*' that $0.01 < p \leq 0.05$ '.' that $0.05 < p \leq 0.10$.
triangles	A character indicating whether the plot should include the lower, upper or both traingle(s). Here it is only used to adjust gridlines for the omission of the diagonal.
title	A character string giving the main title for the plot.
axis.labels	A character string giving the label to use for both the x- and y-axis.
colours	A vector of of colours to be passed to the ggplot function scale_colour_gradientn.
ggplotFuncs	A [list](), each element of which contains the results of evaluating a [ggplot]() function. It is created by calling the [list]() function with a [ggplot]() function call for each element. It is passed to ggplot.
...	Provision for passssing arguments to functions called internally - not used at present.

Value

No values are returned, but a plot is printed.

Author(s)

Chris Brien

See Also

[plotPvalues.alldiffs](), [allDifferences.data.frame](), ggplot

Examples

```
## Not run:
data(WaterRunoff.dat)
asreml.options(keep.order = TRUE) #required for asreml4 only
current.asr <- asreml(fixed = pH ~ Benches + (Sources * (Type + Species)),
                      random = ~ Benches:MainPlots,
                      keep.order=TRUE, data= WaterRunoff.dat)
current.asrt <- asrtests(current.asr, NULL, NULL)
```

```
diffs <- predictPlus.asreml(classify = "Sources:Type",
                            asreml.obj = current.asr, tables = "none",
                            wald.tab = current.asrt$wald.tab,
                            present = c("Type","Species","Sources"))

p <- within(reshape::melt(diffs$p.differences),
            {
              X1 <- factor(X1, levels=dimnames(diffs$p.differences)[[1]])
              X2 <- factor(X2, levels=levels(X1))
            })
names(p)[match("value", names(p))] <- "p"
plotPvalues(p, x = "X1", y = "X2",
            gridspacing = rep(c(3,4), c(4,2)), show.sig = TRUE)

## End(Not run)
```

---

plotVariofaces.data.frame

Plots empirical variogram faces, including envelopes, from supplied residuals as described by Stefanova, Smith & Cullis (2009).

Description

Produces a plot for each face of an empirical 2D variogram based on supplied residuals from both an observed data set and simulated data sets. Those from simulated data sets are used to produce confidence envelopes If the data consists of sections, such as separate experiments, the two variogram faces are produced for each section. This function is less efficient in storage terms than `variofaces.asreml`, because here the residuals from all simulated data sets must be saved, in addition to the values for the variogram faces; in `variofaces.asreml`, the residuals for each simulated data set are discarded after the variogram has been calculated. On the other hand, the present function is more flexible, because there is no restriction on how the residuals are obtained.

Usage

```
## S3 method for class 'data.frame'
plotVariofaces(data, residuals, restype="Residuals", ...)
```

Arguments

data	A `data.frame` with either 3 or 4 columns. Only if there are 4 columns, the first should be a factor indexing sections for which separate variogram plots are to be produced. In either case, the other 3 columns should be, in order, (i) a factor indexing the x-direction, (ii) a factor indexing the y-direction, and (iii) the residuals for the observed response.
residuals	A `data.frame`, with either 2 or 3 initial columns followed by columns, each of which are the residuals from a simulated data set.
restype	A `character` describing the type of residuals that have been supplied. It will be used in the plot titles.
...	Other arguments that are passed down to the function `asreml.variogram`.

Details

For each set of residuals, `asreml.variogram` is used to obtain the empirical variogram, from which the values for its faces are obtained. Plots are produced for each face and include the observed residuals and the 2.5%, 50% & 97.5% quantiles.

Value

A `list` with the following components:

1. **face1:** a `data.frame` containing the variogram values on which the plot for the first dimension is based.

2. **face2:** a `data.frame` containing the variogram values on which the plot for the second dimension is based.

Author(s)

Chris Brien

References

Stefanova, K. T., Smith, A. B. & Cullis, B. R. (2009) Enhanced diagnostics for the spatial analysis of field trials. *Journal of Agricultural, Biological, and Environmental Statistics*, **14**, 392–410.

See Also

asremlPlus-package, asreml, asreml.variogram, variofaces.asreml, simulate.asreml.

Examples

```
## Not run:
data(Wheat.dat)
current.asr <- asreml(yield ~ Rep + WithinColPairs + Variety,
                      random = ~ Row + Column + units,
                      residual = ~ ar1(Row):ar1(Column),
                      data=Wheat.dat)
current.asrt <- asrtests(current.asr, NULL, NULL)
current.asrt <- rmboundary.asrtests(current.asrt)
# Form variance matrix based on estimated variance parameters
s2 <- current.asr$sigma2
gamma.Row <- current.asr$gammas[1]
gamma.unit <- current.asr$gammas[2]
rho.r <- current.asr$gammas[4]
rho.c <- current.asr$gammas[5]
row.ar1 <- mat.ar1(order=10, rho=rho.r)
col.ar1 <- mat.ar1(order=15, rho=rho.c)
V <- gamma.Row * fac.sumop(Wheat.dat$Row) +
  gamma.unit * diag(1, nrow=150, ncol=150) +
  mat.dirprod(col.ar1, row.ar1)
V <- s2*V

#Produce variogram faces plot (Stefanaova et al, 2009)
resid <- simulate(current.asr, V=V, which="residuals")
resid$residuals <- cbind(resid$observed[c("Row","Column")],
                         resid$residuals)
plotVariofaces(data=resid$observed[c("Row","Column","residuals")],
```

```
                    residuals=resid$residuals,
                    restype="Standardized conditional residuals")

## End(Not run)
```

---

powerTransform	*Performs a combination of a linear and a power transformation on a variable. The transformed variable is stored in the* data.frame data.

Description

Perform a combination of a linear and a power transformation on a variable whose name is given as a character string in var.name. The transformed variable is stored in the data.frame data. The name of the transformed variable is made by prepending to the original var.name a combination of (i) .offset, if offset is nonzero, (ii) neg., if scale is -1, or scaled., if abs(scale) is other than one, and (iii) either log., sqrt., recip. or power., if power is other than one. No action is taken if there is no transformation (i.e. offset = 0, scale = 1 and power = 1). Also, the titles list is extended to include a component with a generated title for the transformed variable with text indicating the transformation prepended to the title for the var.name obtained from the titles list. For nonzero offset, 'Offset ' is prepended, For scaled not equal to one, the possible prepends are 'Negative of ' and 'Scaled '. The possible prepended texts for power not equal to one are 'Logarithm of', 'Square root of ', 'Reciprocal of ' and 'Power nnnn of ', where nnn is the power used.

Usage

```
powerTransform(var.name, power = 1, offset = 0, scale = 1, titles = NULL, data)
```

Arguments

var.name	A character string specifying the name of the variable in the data.frame data that is to be transformed.
power	A number specifying the power to be used in the transformation. If equal to 1, the default, no power transformation is applied. Otherwise, the variable is raised to the specified power, after scaling and applying any nonzero offset. If power = 0, the natural logarithm is used to transform the response; however, if the smallest value to be log-transformed is less than 1e-04, an error is generated. A log-transformation in this situation may be possible if a nonzero offset and/or a scale not equal to one is used.
offset	A number to be added to each value of the variable, after any scaling and before applying any power transformation.
scale	A number to multiply each value of the variable, before adding any offset and applying any power transformation.
titles	A character vector, each element of which is named for a variable in data and is a character string giving a title to use in output (e.g. tables and graphs) involving the variable. If titles are not supplied, the column name of the variable in data is used.
data	A data.frame containing the variable to be transformed and to which the transformed variable is to be appended.

Value

A `list` with a component named `data` that is the `data.frame` containing the transformed variable, a component named tvar.name that is a `character string` that is the name of the transformed variable in `data`, and a component named `titles` that extends the `list` supplied in the `titles` argument to include a generated title for the transformed title, the name of the new component being `tvar.name`.

Author(s)

Chris Brien

See Also

`angular`, `angular.mod`.

Examples

```
## set up a factor with labels
x.dat <- data.frame(y = c(14, 42, 120, 150))

## transform y to logarithms
trans <- powerTransform("y", power = 0, titles=list(y = "Length (cm)"), data = x.dat)
x.dat <- trans$data
tvar.name <- trans$tvar.name

## transform y to logarithms after multiplying by -1 and adding 1.
z.dat <- data.frame( y = c(-5.25, -4.29, -1.22, 0.05))
trans <- powerTransform("y", power = 0, scale = -1, offset = 1 ,
                        titles=list(y = "Potential"), data = z.dat)
z.dat <- trans$data
tvar.name <- trans$tvar.name
```

predictPlus.asreml	*Forms the predictions and associated statistics for a term, using an asreml object and a wald.tab and taking into account that a numeric vector and a factor having parallel values may occur in the model. If a linear transformation is specified then the values of the linear transformation of predictions are returned, along with their standard errors and the pairwise differences and associated statistics. It stores the results in an object of class alldifffs and may print the results. It can be used when there are not parallel values.*

Description

This function forms the predictions for `term` using `classify` and the supplied `asreml` object and stores them in an `alldiffs` object. If `x.num` is supplied, the predictions will be obtained for the values supplied in `x.pred.values` and, if supplied, `x.plot.values` will replace them in the `alldiffs` object that is returned. If `x.fac`, but not `x.num`, is specified, predictions will involve it and, if supplied, `x.plot.values` will replace the levels of `x.fac` in the `alldiffs` object that is returned. In order to get the correct predictions you may need to supply additional arguments to `predict.asreml` through `...` e.g. `present`, `parallel`, `levels`. Any aliased predictions will be removed, as will any standard error of pairwise differences involving them.

Also calculated are the approximate degrees of freedom of the standard errors of the predictions. If the deominator degrees of freedom for `term` are available in `wald.tab`, they are used. Otherwise the residual degrees of freedom or the maximum of the denominator degrees in `wald.tab`, excluding the Intercept, are used. Which is used depends on the setting of `dDF.na`. These degrees of freedom are used for the t-distribution on which p-values and confidence intervals are based. It is stored as an attribute to the `alldiffs` object. The degrees of freedom are also used in calculating the minimum, mean and maximum LSD for comparing pairs of predictions, which are also stored in the `alldiffs` object.

If `pairwise = TRUE`, all pairwise differences between the `predictions`, their standard errors, p-values and LSD statistics are computed using [allDifferences.data.frame](). This adds them to the `alldiffs` object as additional `list` components named `differences`, `sed`, `p.differences` and `LSD`.

If a transformation has been applied (any one of `transform.power` is not one, `scale` is not one and `offset` is nonzero), the backtransforms of the transformed values and their lower and upper confidence intervals are added to a `data.frame` that is consistent with an object of class `asremlPredict`, such as is stored in the `pvals` component of the `prediction` component of the value produced by `predict.asreml`. If `transform.power` is other than one, the `standard.error` column of the `data.frame` is set to `NA`. This `data.frame` is added to the `alldiffs` object as a `list` component called `backtransforms`.

The printing of the components produced is controlled by the `tables` argument. The order of plotting the levels of one of the factors indexing the predictions can be modified and is achieved using [sort.alldiffs]().

Usage

```
## S3 method for class 'asreml'
predictPlus(asreml.obj, classify, term = NULL,
            linear.transformation = NULL, titles = NULL,
            x.num = NULL, x.fac = NULL,
            x.pred.values = NULL, x.plot.values = NULL,
            error.intervals = "Confidence", avsed.tolerance = 0.25,
            meanLSD.type = "overall", LSDby = NULL,
            pairwise = TRUE, Vmatrix = FALSE,
            tables = "all" , level.length = NA,
            transform.power = 1, offset = 0, scale = 1,
            inestimable.rm = TRUE,
            sortFactor = NULL, sortWithinVals = NULL,
            sortOrder = NULL, decreasing = FALSE,
            wald.tab = NULL, alpha = 0.05,
            dDF.na = "residual",  dDF.values = NULL,
            trace = FALSE, ...)
```

Arguments

asreml.obj	asreml object for a fitted model.
classify	a [character]() string giving the variables that define the margins of the multiway table to be predicted. Multiway tables are specified by forming an interaction type term from the classifying variables, that is, separating the variable names with the : operator.
term	a [character]() string giving the variables that define the term that was fitted using asreml and that corresponds to classify. It only needs to be specified when it is different to classify.

linear.transformation

a [formula](#) or a [matrix](#). If a [formula](#) is given then it is taken to be a submodel of the model term corresponding to the classify. The projection matrix that transforms the predictions so that they conform to the submodel is obtained; the submodel should involving the variables in the classify. For example, for classify set to "A:B", the submodel ~ A + B will result in the predictions for the combinations of A and B being made additive for the [factor](#)s A and B.

If a [matrix](#) is provided then it will be used to apply the linear transformation to the predictions. It might be a contrast [matrix](#) or a [matrix](#) of weights for a factor used to obtain the weighted average over that factor. The number of rows in the [matrix](#) should equal the number of linear combinations of the predictions desired and the number of columns should equal the number of predictions.

In either case, as well as the values of the linear combinations, their standard errors, pairwise differences and associated statistics are returned.

titles
A [list](#), each component of which is named for a column in the data.frame for asreml.obj and contains a character string giving a title to use in output (e.g. tables and graphs). Here they will be used for table headings.

x.num
A [character](#) string giving the name of the numeric covariate that (i) corresponds to x.fac, (ii) is potentially included in terms in the fitted model, and (iii) which corresponds to the x-axis variable. It should have the same number of unique values as the number of levels in x.fac.

x.fac
A [character](#) string giving the name of the factor that (i) corresponds to x.num, (ii) is potentially included in terms in the fitted model, and (iii) which corresponds to the x-axis variable. It should have the same number of levels as the number of unique values in x.num. The levels of x.fac must be in the order in which they are to be plotted - if they are dates, then they should be in the form yyyymmdd, which can be achieved using as.Date. However, the levels can be non-numeric in nature, provided that x.num is also set.

x.pred.values
The values of x.num for which predicted values are required. If levels is set for passing to predict.asreml, x.pred.values is ignored. Note that while levels is and alternative to x.pred.values, it allows more general setting of the levels to be predicted.

x.plot.values
The actual values to be plotted on the x axis. They are needed when values different to those in x.num are to be plotted or x.fac is to be plotted because there is no x.num term corresponding to the same term with x.fac.

error.intervals

A [character](#) string indicating the type of error interval, if any, to calculate in order to indicate uncertainty in the results. Possible values are "none", "StandardError", "Confidence" and "halfLeastSignificant". The default is for confidence limits to be used. The "halfLeastSignificant" option results in half the Least Significant Difference (LSD) being added and subtracted to the predictions, the LSD being calculated using the square root of the mean of the variances of all or a subset of pairwise differences between the predictions. If meanLSD.type is set to overall, the avsed.tolerance is not NA and the range of the SEDs divided by the average of the SEDs exceeds avsed.tolerance then the error.intervals calculations and the plotting will revert to confidence intervals. Also, half LSDs cannot be used for backtansformed values and so confidence intervals will be used instead.

avsed.tolerance

A [numeric](#) giving the value of the SED range, the range of the SEDs divided by the square root of the mean of the variances of all or a subset of the pairwise

differences, that is considered reasonable in calculating error.intervals. It should be a value between 0 and 1. The following rules apply:

1. If avsed.tolerance is NA then mean LSDs of the type specified by meanLSD.type are calculated and used in error.intervals and plots.

2. Irrespective of the setting of meanLSD.type, if avsed.tolerance is not exceeded then the mean LSDs are used in error.intervals and plots.

3. If meanLSD.type is set to overall, avsed.tolerance is not NA, and avsed.tolerance is exceeded then error.intervals and plotting revert to confidence intervals.

4. If meanLSD.type is set to factor.combinations and avsed.tolerance is not exceeded for any factor combination then the half LSDs are used in error.intervals and plots; otherwise, error.intervals and plotting revert to confidence intervals.

5. If meanLSD.type is set to per.prediction and avsed.tolerance is not exceeded for any prediction then the half LSDs are used in error.intervals and plots; otherwise, error.intervals and plotting revert to confidence intervals.

meanLSD.type A [character](#) string determining whether the mean LSD stored is (i) the overall mean, based on the square root of the mean of the variances of all pairwise variances, (ii) the mean for each factor.combination of the [factor](#)s specified by LSDby, which is based on the square root of the mean of the variances for all pairwise differences for each factor combination, or (iii) the per.prediction mean, based, for each prediction, on the square root of the mean of the variances for all pairwise differences involving that prediction. It also determines, in conjunction with avsed.tolerance, which LSD will be used in calculating error.intervals and, hence, is used for plots.

LSDby A [character](#) (vector) of factor names, being the names of the [factor](#)s for each of whose levels combinations a mean LSD, minLSD and max LSD is stored in the LSD component of the alldiffs object when meanLSD.type is factor.combinatons.

pairwise A [logical](#) indicating whether all pairwise differences of the predictions and their standard errors and p-values are to be computed and stored. If tables is equal to "differences" or "all" or error.intervals is equal to "halfLeastSignificant", they will be stored irrespective of the value of pairwise.

Vmatrix A [logical](#) indicating whether the variance matrix of the predictions will be stored as a component of the alldiffs object that is returned. If linear.transformation is set, it will be stored irrespective of the value of Vmatrix.

tables A [character](#) vector containing a combination of none, predictions, vcov, backtransforms, differences, p.differences, sed, LSD and all. These nominate which components of the alldiffs object to print.

level.length The maximum number of characters from the the levels of factors to use in the row and column labels of the tables of pairwise differences and their p-values and standard errors.

transform.power

A [numeric](#) specifying the power of a transformation, if one has been applied to the response variable. Unless it is equal to 1, the default, back-transforms of the predictions will be obtained and presented in tables or graphs as appropriate. The back-transformation raises the predictions to the power equal to the reciprocal of transform.power, unless it equals 0 in which case the exponential of the predictions is taken.

offset	A [numeric](#) that has been added to each value of the response after any scaling and before applying any power transformation.
scale	A [numeric](#) by which each value of the response has been multiplied before adding any offset and applying any power transformation.
inestimable.rm	A logical indicating whether rows for predictions that are not estimable are to be removed from the components of the alldiffs object.
sortFactor	A character containing the name of the factor that indexes the set of predicted values that determines the sorting of the alldiffs components by [sort.alldiffs](#). If NULL then sorting is not carried out. If there is more than one variable in the classify term then sortFactor is sorted for the predicted values within each combination of the values of the sortWithin variables: the classify variables, excluding the sortFactor. There should be only one predicted value for each unique value of sortFactor within each set defined by a combination of the values of the sortWithin variables.
sortWithinVals	A list with a component named for each factor and numeric that is a classify variable for the predictions, excluding sortFactor. Each component should contain a single value that is a value of the variable. The combination of this set of values will be used to define a subset of the predicted values whose order will define the order of sortFactor to be used for all combinations of the sortWithinVals variables. If sortWithinVals is NULL then the first value of each sortWithin variable in predictions component is used to define sortWithinVals. If there is only one variable in the classify then sortWithinVals is ignored.
sortOrder	A character vector whose length is the same as the number of levels for sortFactor in the predictions component of the alldiffs object. It specifies the desired order of the levels in the reordered components of the alldiffs object. The argument sortWithinVals is ignored. The following creates a sortOrder vector levs for factor f based on the values in x: levs <- levels(f)[order(x)].
decreasing	A logical passed to order that detemines whether the order for sorting the alldiffs components is for increasing or decreasing magnitude of the predicted values.
wald.tab	A [data.frame](#) containing the pseudo-anova table for the fixed terms produced by a call to wald.asreml. The main use of it here is in determinining the degrees of freedom of the standard errors of the predictions. denominator degrees of freedom when p-values or confidence intervals are to be calculated.
alpha	A [numeric](#) giving the significance level for LSDs or one minus the confidence level for confidence intervals.
dDF.na	A [character](#) specifying the method to use to obtain approximate denominator degrees of freedom. when the numeric or algebraic methods produce an NA. Consistent with when no denDF are available, the default is "residual" and so the residual degrees of freedom from asreml.obj$nedf are used. If dDF.na = "none", no subtitute denominator degrees of freedom are employed; if dDF.na = "maximum", the maximum of those denDF that are available, excluding that for the Intercept, is used; if all denDF are NA, asreml.obj$nedf is used. If dDF.na = "supplied", a vector of values for the denominator degrees of freedom is to be supplied in dDF.values. Any other setting is ignored and a warning message produced. Generally, substituting these degrees of freedom is anticonservative in that it is likely that the degrees of freedom used will be too large.

dDF.values	A [vector](#) of values to be used when dDF.na = "supplied". Its values will be used when denDF in a test for a fixed effect is NA. This vector must be the same length as the number of fixed terms, including (Intercept) whose value could be NA.
trace	A [logical](#) that control output from ASReml-R. If TRUE then partial iteration details are displayed when ASReml-R functions are invoked; if FALSE then no output is displayed.
...	further arguments passed to predict.asreml.

Value

For linear.transformations set to NULL, an object of S3-class alldiffs with predictions and their standard errors and, depending on the settings of the arguments, all pairwise differences between predictions, their standard errors and p-values and LSD statistics. Also, unless the sortFactor or sortOrder arguments are invoked, the rows of predictions component are ordered so that they are in standard order for the variables in the classify. That is, the values of the last variable change with every row, those of the second-last variable only change after all the values of the last variable have been traversed; in general, the values of a variable are the same for all the combinations of the values to the variables to its right in the classify. In addition, if necessary, the order of the columns of the variables in the predictions component are changed to match their order in the classify.

If transform.power or scale is not one or offset is not zero, it will contain a data.frame with the backtransformed linear transformation of the predictions. The backtransformation will, after backtransforming for any power transformation, subtract the offset and then divide by the scale.

If error.intervals is not "none", then the predictions component and, if present, the backtransforms component will contain columns for the lower and upper values of the limits for the interval. The names of these columns will consist of three parts separated by full stops: 1) the first part will be lower or upper; 2) the second part will be one of Confidence, StandardError or halfLeastSignificant; 3) the third component will be limits.

The name of the response, the response.title, the term, the classify, tdf, sortFactor and the sortOrder will be set as attributes to the object. Note that the classify in an alldiffs object is based on the variables indexing the predictions, which may differ from the classify used to obtain the original predictions (for example, when the alldiffs objects stores a linear transformation of predictions.

For linear.transformations set to other than NULL, an alldiffs object with the linear.transformation applied to the predictions and their standard errors and, depending on the settings of the arguments, all pairwise differences between the linearly transformed predictions, their standard errors and p-values and LSD statistics. (See also [linTransform.alldiffs](#).)

Author(s)

Chris Brien

See Also

[as.alldiffs](#), [print.alldiffs](#), [linTransform.alldiffs](#), [sort.alldiffs](#), [subset.alldiffs](#), [allDifferences.data.frame](#), [redoErrorIntervals.alldiffs](#), [recalcLSD.alldiffs](#), [predictPresent.asreml](#), [plotPredictions.data.frame](#), as.Date, predict.asreml

Examples

```
## Not run:
data(WaterRunoff.dat)
asreml.options(keep.order = TRUE) #required for asreml4 only
current.asr <- asreml(fixed = pH ~ Benches + (Sources * (Type + Species)),
                        random = ~ Benches:MainPlots,
                        keep.order=TRUE, data= WaterRunoff.dat)
current.asrt <- asrtests(current.asr, NULL, NULL)
diffs <- predictPlus(classify = "Sources:Type",
                        asreml.obj = current.asr,
                        wald.tab = current.asrt$wald.tab,
                        present = c("Sources", "Type", "Species"))

## End(Not run)
```

---

predictPresent.asreml *Forms the predictions for each of one or more terms and presents them in tables and/or graphs.*

Description

This function forms the predictions for each term in terms using a supplied asreml object and [predictPlus.asreml](). Tables are produced using [predictPlus.asreml](), in conjunction with [allDifferences.data.f]() with the argument tables specifying which tables are printed. The argument plots, along with transform.power, controls which plots are produced. The plots are produced using [plotPredictions.data.frame](), with line plots produced when variables involving x.num or x.fac are involved in classify for the predictions and bar charts otherwise. In order to get the correct predictions you may need to supply additional arguments to predict.asreml through ... e.g. present, parallel, levels.

The order of plotting the levels of one of the factors indexing the predictions can be modified and is achieved using [sort.alldiffs]().

Usage

```
## S3 method for class 'asreml'
predictPresent(asreml.obj, terms,
                linear.transformation = NULL,
                wald.tab = NULL, dDF.na = "residual", dDF.values = NULL,
                x.num = NULL, x.fac = NULL, nonx.fac.order = NULL,
                x.pred.values = NULL, x.plot.values = NULL,
                plots = "predictions", panels = "multiple",
                graphics.device = NULL,
                error.intervals = "Confidence", meanLSD.type = "overall",
                LSDby = NULL, avsed.tolerance = 0.25, titles = NULL,
                colour.scheme = "colour", save.plots = FALSE,
                transform.power = 1, offset = 0, scale = 1,
                pairwise = TRUE, Vmatrix = FALSE,
                tables = "all", level.length = NA,
                alpha = 0.05, inestimable.rm = TRUE,
                sortFactor = NULL, sortWithinVals = NULL,
                sortOrder = NULL, decreasing = FALSE,
                trace = FALSE, ggplotFuncs = NULL, ...)
```

Arguments

asreml.obj   asreml object for a fitted model.

terms   a character vector giving the terms for which predictions are required.

linear.transformation

a [formula](#) or a [matrix](#) specifying a linear transformation to be applied to the predictions. If a [formula](#) is given then it is taken to be a submodel of the model term corresponding to the classify. The projection matrix that transforms the predictions so that they conform to the submodel is obtained; the submodel should involving the variables in the classify. For example, for classify set to "A:B", the submodel ~ A + B will result in the predictions for the combinations of A and B being made additive for the [factor](#)s A and B.

If a [matrix](#) is provided then it will be used to apply the linear transformation to the predictions. It might be a contrast [matrix](#) or a [matrix](#) of weights for a factor used to obtain the weighted average over that factor. The number of rows in the [matrix](#) should equal the number of linear combinations of the predictions desired and the number of columns should equal the number of predictions.

In either case, as well as the values of the linear combinations, their standard errors, pairwise differences and associated statistics are returned in the alldiffs object.

wald.tab   a data frame containing the pseudo-anova table for the fixed terms produced by a call to wald.asreml. The main use of it here is in getting denominator degrees of freedom when confidence intervals are to be plotted.

dDF.na   the method to use to obtain approximate denominator degrees of freedom. when the numeric or algebraic methods produce an NA. Consistent with when no denDF are available, the default is "residual" and so the residual degrees of freedom from asreml.obj$nedf are used. If dDF.na = "none", no subtitute denominator degrees of freedom are employed; if dDF.na = "maximum", the maximum of those denDF that are available, excluding that for the Intercept, is used; if all denDF are NA, asreml.obj$nedf is used. If dDF.na = "supplied", a vector of values for the denominator degrees of freedom is to be supplied in dDF.values. Any other setting is ignored and a warning message produced. Generally, substituting these degrees of freedom is anticonservative in that it is likely that the degrees of freedom used will be too large.

dDF.values   A vector of values to be used when dDF.na = "supplied". Its values will be used when denDF in a test for a fixed effect is NA. This vector must be the same length as the number of fixed terms, including (Intercept) whose value could be NA.

x.num   A character string giving the name of the numeric covariate that corresponds to x.fac, is potentially included in terms in the fitted model and which corresponds to the x-axis variable. It should have the same number of unique values as the number of levels in x.fac.

x.fac   A character string giving the name of the factor that corresponds to x.num, is potentially included in terms in the fitted model and which corresponds to the x-axis variable. It should have the same number of levels as the number of unique values in x.num. The levels of x.fac must be in the order in which they are to be plotted - if they are dates, then they should be in the form yyyymmdd, which can be achieved using as.Date. However, the levels can be non-numeric in nature, provided that x.num is also set.

nonx.fac.order   A character vector giving the order in which factors other than x.fac are to be plotted in plots with multiple panels (i.e. where the number of non-x factors is greater than 1). The first factor in the vector will be plotted on the X axis (if there is no x.num or x.fac. Otherwise, the order of plotting the factors is in columns (X facets) and then rows (Y facets). By default the order is in decreasing order for the numbers of levels of the non x factors.

x.pred.values   The values of x.num for which predicted values are required.

x.plot.values   The actual values to be plotted on the x axis or in the labels of tables. They are needed when values different to those in x.num are to be plotted or x.fac is to be plotted because there is no x.num term corresponding to the same term with x.fac.

plots   Possible values are "none", "predictions", "backtransforms" and "both". Plots are not produced if the value is "none". If data are not transformed for analysis (transform.power = 1), a plot of the predictions is produced provided plots is not "none". If the data are transformed, the value of plots determines what is produced.

panels   Possible values are "single" and "multiple". When line plots are to be produced, because variables involving x.num or x.fac are involved in classify for the predictions, panels determines whether or not a single panel or multiple panels in a single window are produced. The panels argument is ignored for for bar charts.

graphics.device
                 A character specifying a graphics device for plotting. The default is graphics.device = NULL, which will result in plots being produced on the current graphics device. Setting it to "windows", for example, will result in a windows graphics device being opened.

error.intervals
                 A character string indicating the type of error interval, if any, to calculate in order to indicate uncertainty in the results. Possible values are "none", "StandardError", "Confidence" and "halfLeastSignificant". The default is for confidence limits to be used. The "halfLeastSignificant" option results in half the Least Significant Difference (LSD) being added and subtracted to the predictions, the LSD being calculated using the square root of the mean of the variances of all or a subset of pairwise differences between the predictions. If meanLSD.type is set to overall, the avsed.tolerance is not NA and the range of the SEDs divided by the average of the SEDs exceeds avsed.tolerance then the error.intervals calculations and the plotting will revert to confidence intervals. Also, half LSDs cannot be used for backtansformed values and so confidence intervals will be used instead.

avsed.tolerance
                 A numeric giving the value of the SED range, the range of the SEDs divided by the square root of the mean of the variances of all or a subset of the pairwise differences, that is considered reasonable in calculating error.intervals. It should be a value between 0 and 1. The following rules apply:

   1. If avsed.tolerance is NA then mean LSDs of the type specified by meanLSD.type are calculated and used in error.intervals and plots.
   2. Irrespective of the setting of meanLSD.type, if avsed.tolerance is not exceeded then the mean LSDs are used in error.intervals and plots.
   3. If meanLSD.type is set to overall, avsed.tolerance is not NA, and avsed.tolerance is exceeded then error.intervals and plotting revert to confidence intervals.

4. If meanLSD.type is set to factor.combinations and avsed.tolerance is not exceeded for any factor combination then the half LSDs are used in error.intervals and plots; otherwise, error.intervals and plotting revert to confidence intervals.

5. If meanLSD.type is set to per.prediction and avsed.tolerance is not exceeded for any prediction then the half LSDs are used in error.intervals and plots; otherwise, error.intervals and plotting revert to confidence intervals.

meanLSD.type   A character string determining whether the mean LSD stored is (i) the overall mean, based on the square root of the mean of the variances of all pairwise variances, (ii) the mean for each factor.combination of the factors specified by LSDby, which is based on the square root of the mean of the variances for all pairwise differences for each factor combination, or (iii) the per.prediction mean, based, for each prediction, on the square root of the mean of the variances for all pairwise differences involving that prediction. It also determines, in conjunction with avsed.tolerance, which LSD will be used in calculating error.intervals and, hence, is used for plots.

LSDby   A character (vector) of factor names, being the names of the factors for each of whose levels combinations a mean LSD, minLSD and max LSD is stored in the LSD component of the alldiffs object when meanLSD.type is factor.combinatons.

titles   A list, each component of which is named for a column in the data.frame for asreml.obj and contains a character string giving a title to use in output (e.g. tables and graphs). Here they will be used for axis labels.

colour.scheme   A character string specifying the colour scheme for the plots. The default is "colour" which produces coloured lines and bars, a grey background and white gridlines. A value of "black" results in black lines, grey bars and gridlines and a white background.

save.plots   A logical that determines whether any plots will be saved. If they are to be saved, a file name will be generated that consists of the following elements separated by full stops: the response variable name with .back if backtransformed values are being plotted, the classify term, Bar or Line and, if error.intervals is not "none", one of SE, CI or LSI. The file will be saved as a 'png' file in the current work directory.

transform.power

A number specifying the power of a transformation, if one has been applied to the response variable. Unless it is equal to 1, the default, back-transforms of the predictions will be obtained and presented in tables or graphs as appropriate. The back-transform will raise the predictions to the power equal to the reciprocal of transform.power, unless it equals 0 in which case the exponential will be taken. Any scaling and offsetting will also be taken into account in the backtransformation.

offset   A number that has been added to each value of the response after any scaling and before applying any power transformation. Unless it is equal to 0, the default, back-transforms of the predictions will be obtained and presented in tables or graphs as appropriate. The backtransformation will, after backtransforming for any power transformation, subtract the offset.

scale   A number by which each value of the response has been multiply before adding any offset and applying any power transformation. Unless it is equal to 1, the default, back-transforms of the predictions will be obtained and presented in

	tables or graphs as appropriate. The backtransformation will, after backtransforming for any power transformation and then subtracting the offset, divide by the scale.
pairwise	A logical indicating whether all pairise differences of the predictions and their standard errors and p-values are to be computed and stored. If tables is equal to "differences" or "all" or error.intervals is equal to "halfLeastSignificant", they will be stored irrespective of the value of pairwise.
Vmatrix	A [logical](#) indicating whether the variance matrix of the predictions will be stored as a component of the alldiffs object that is returned. If linear.transformation is set, it will be stored irrespective of the value of Vmatrix.
tables	A character vector containing a combination of predictions, vcov, backtransforms, differences, p.differences, sed, LSD and all. These nominate which components of the alldiffs object to print.
level.length	The maximum number of characters from the the levels of factors to use in the row and column labels of the tables produced by [allDifferences.data.frame](#).
alpha	the significance level for LSDs or 1 - alpha is the confidence level for confidence intervals.
inestimable.rm	A logical indicating whether rows for predictions that are not estimable are to be removed from the components of the alldiffs object.
sortFactor	A character containing the name of the factor that indexes the set of predicted values that determines the sorting of the alldiffs components by [sort.alldiffs](#). If NULL then sorting is not carried out. If there is more than one variable in the classify term then sortFactor is sorted for the predicted values within each combination of the values of the sortWithin variables: the classify variables, excluding the sortFactor. There should be only one predicted value for each unique value of sortFactor within each set defined by a combination of the values of the sortWithin variables.
sortWithinVals	A list with a component named for each factor and numeric that is a classify variable for the predictions, excluding sortFactor. Each component should contain a single value that is a value of the variable. The combination of this set of values will be used to define a subset of the predicted values whose order will define the order of sortFactor to be used for all combinations of the sortWithinVals variables. If sortWithinVals is NULL then the first value of each sortWithin variable in predictions component is used to define sortWithinVals. If there is only one variable in the classify then sortWithinVals is ignored.
sortOrder	A character vector whose length is the same as the number of levels for sortFactor in the predictions component of the alldiffs object. It specifies the desired order of the levels in the reordered components of the alldiffs object. The argument sortWithinVals is ignored. The following creates a sortOrder vector levs for factor f based on the values in x: levs <- levels(f)[order(x)].
decreasing	A logical passed to order that detemines whether the order for sorting the alldiffs components is for increasing or decreasing magnitude of the predicted values.
trace	if TRUE then partial iteration details are displayed when ASReml-R functions are invoked; if FALSE then no output is displayed.
ggplotFuncs	A [list](#), each element of which contains the results of evaluating a [ggplot](#) function. It is created by calling the [list](#) function with a [ggplot](#) function call for each element. It is passed to [plotPredictions.data.frame](#).

...	further arguments passed to predict.asreml via predictPlus.asreml and to
	ggplot via plotPredictions.data.frame.

Value

a list containing an alldiffs object for each term for which tables are produced. The names of
the components of this list are the terms with full-stops (.) replacing colons (:). Plots are also
preduced depending on the setting of the plot argument.

Author(s)

Chris Brien

See Also

predictPlus.asreml, allDifferences.data.frame, sort.alldiffs, subset.alldiffs,
redoErrorIntervals.alldiffs, recalcLSD.alldiffs, plotPredictions.data.frame,
print.alldiffs, as.Date, Devices

Examples

```
## Not run:
data(WaterRunoff.dat)
titles <- list("Days since first observation", "Days since first observation",
               "pH", "Turbidity (NTU)")
names(titles) <- names(WaterRunoff.dat)[c(5,7,11:12)]
asreml.options(keep.order = TRUE) #required for asreml4 only
current.asr <- asreml(fixed = log.Turbidity ~ Benches + Sources + Type + Species +
                                    Sources:Type + Sources:Species + Sources:Species:xDay +
                                    Sources:Species:Date,
                      data = WaterRunoff.dat, keep.order = TRUE)
current.asrt <- asrtests(current.asr, NULL, NULL)

#### Get the observed combinations of the factors and variables in classify
class.facs <- c("Sources","Species","Date","xDay")
levs <- as.data.frame(table(WaterRunoff.dat[class.facs]))
levs <- levs[do.call(order, levs), ]
levs <- as.list(levs[levs$Freq != 0, class.facs])
levs$xDay <- as.numfac(levs$xDay)

#### parallel and levels are arguments from predict.asreml
diff.list <- predictPresent.asreml(asreml.obj = current.asrt$asreml.obj,
                                   terms = "Date:Sources:Species:xDay",
                                   x.num = "xDay", x.fac = "Date",
                                   parallel = TRUE, levels = levs,
                                   wald.tab = current.asrt$wald.tab,
                                   plots = "predictions",
                                   error.intervals = "StandardError",
                                   titles = titles,
                                   transform.power = 0,
                                   present = c("Type","Species","Sources"),
                                   tables = "none",
                                   level.length = 6)

## End(Not run)
```

---

print.alldiffs   *Prints the values in an alldiffs object in a nice format.*

Description

Prints the predictions and standard errors from using asreml to fit models in the same way as asreml prints them. Also prints out all pairwise differences between the predictions to 2 signigicant figures, along with their p-values and standard errors to 4 decimal places. If LSDs are requested the mean, minimum and maximum LSDs will be printed.

Usage

```
## S3 method for class 'alldiffs'
print(x, which = "all", ...)
```

Arguments

x	An object of S3-class alldiffs.
which	A character vector containing a combination of predictions, vcov, backtransforms, differences, p.differences, sed, LSD and all. These nominate which components of the alldiffs object to print.
...	further arguments passed to or from other methods.

Value

No value is returned, but the components of x are printed.

Author(s)

Chris Brien

See Also

[as.alldiffs](), [allDifferences.data.frame]()

Examples

```
## Not run:
print.alldiffs(diffs, which = "predictions")

## End(Not run)
```

print.asrtests	*Prints the values in an* asrtests *object*

Description

Prints a summary of the asreml object and the test.summary data.frame that are stored in the asrtests object.

Usage

```
## S3 method for class 'asrtests'
print(x, which = "all", ...)
```

Arguments

x   An asrtests object.

which   Which elements of the asrtests object to print. Possible values are some com-
             bination of asremlsummary, pseudoanova, testsummary and all.

...   further arguments passed to print.

Value

No value is returned, but the elements of the list in x are printed.

Author(s)

Chris Brien

See Also

asrtests, asremlPlus-package

Examples

```
## Not run:
data(Wheat.dat)
current.asr <- asreml(yield ~ Rep + WithinColPairs + Variety,
                      random = ~ Row + Column + units,
                      residual = ~ ar1(Row):ar1(Column),
                      data=Wheat.dat)
current.asrt <- asrtests(current.asr, NULL, NULL)
current.asrt <- rmboundary.asrtests(current.asrt)
# Test Row autocorrelation
current.asrt <- testresidual(current.asrt, , "~ Row:ar1(Column)",
                             label="Row autocorrelation", simpler=TRUE)
print(current.asrt)

## End(Not run)
```

recalcLSD.alldiffs   *Adds or recalculates the LSD component of an alldiffs object.*

Description

Given an alldiffs object, adds or recalculate its LSD component.

Usage

```
## S3 method for class 'alldiffs'
recalcLSD(alldiffs.obj, meanLSD.type = "overall", LSDby = NULL,
          alpha = 0.05, ...)
```

Arguments

alldiffs.obj	An object of S3-class alldiffs (see `as.alldiffs`).
meanLSD.type	A `character` string determining whether the mean LSD stored is (i) the overall mean, based on the square root of the mean of the variances of all pairwise variances, (ii) the mean for each factor.combination of the `factors` specified by LSDby, which is based on the square root of the mean of the variances for all pairwise differences for each factor combination, or (iii) the per.prediction mean, based, for each prediction, on the square root of the mean of the variances for all pairwise differences involving that prediction. It also determines, in conjunction with avsed.tolerance, which LSD will be used in calculating error.intervals and, hence, is used for plots.
LSDby	A `character` (vector) of factor names, being the names of the `factors` for each of whose levels combinations a mean LSD, minLSD and max LSD is stored in the LSD component of the alldiffs object when meanLSD.type is factor.combinatons.
alpha	The significance level for an LSD to compare a pair of predictions.
...	further arguments passed to `allDifferences.data.frame`.

Value

An `alldiffs` object that is a list with components predictions containing the predictions and their standard errors, vcov containing the variance matrix of the predictions, differences containing all pairwise differences between the predictions, p.differences containing p-values for all pairwise differences between the predictions, sed containing the standard errors of all pairwise differences between the predictions, and an LSD containing the mean, minimum and maximum LSDs.

Author(s)

Chris Brien

See Also

asremlPlus-package, as.alldiffs, sort.alldiffs, subset.alldiffs, print.alldiffs, reorderClassify.alldiffs, redoErrorIntervals.alldiffs, plotPredictions.data.frame, predictPlus.asreml, predictPresent.asreml

Examples

```
## Not run:
data(WaterRunoff.dat)
asreml.options(keep.order = TRUE) #required for asreml4 only
current.asr <- asreml(fixed = pH ~ Benches + (Sources * (Type + Species)),
                      random = ~ Benches:MainPlots,
                      keep.order=TRUE, data= WaterRunoff.dat)
current.asrt <- asrtests(current.asr, NULL, NULL)
diffs <- predictPlus(classify = "Sources:Type",
                     asreml.obj = current.asr,
                     wald.tab = current.asrt$wald.tab,
                     present = c("Sources", "Type", "Species"))
diffs <- recalcLSD.alldiffs(diffs, meanLSD.type = "factor.combinations", LSDby = "Sources")

## End(Not run)
```

---

recalcWaldTab.asrtests

Recalculates the denDF, F.inc and P values for a table of Wald test statistics obtained using wald.asreml

---

Description

If some or all denDF are not available, either because they are NA or because F.inc values were not calculated, this function allows the user to specify how approximate denDF values are to be obtained. This is done through the dDF.na and dDF.values arguments. Note that if denDF values are available in the Wald table then only those that are NA will be replaced. It is noted that, as of asreml version 4, wald.asreml has a codekenadj argument.

Usage

```
## S3 method for class 'asrtests'
recalcWaldTab(asrtests.obj, recalc.wald = FALSE,
              denDF="numeric", dDF.na = "none",
              dDF.values = NULL, trace = FALSE, ...)
```

Arguments

asrtests.obj	an [asrtests](#) object for a fitted model that is a list containing an asreml object, a wald.tab data.frame with 4 columns, and a data.frame with 5 columns that records any previous changes and tests in the fitted model.
recalc.wald	a logical indicating whether to call wald.asreml to recalculate the pseudoanova table for the model fit stored in the asreml object contained in [asrtests](#).
denDF	Specifies the method to use in computing approximate denominator degrees of freedom when wald.asreml is called. Can be none to suppress the computations, numeric for numerical methods, algebraic for algebraic methods or default, the default, to autommatically choose numeric or algebraic computations depending on problem size. The denominator degrees of freedom are calculated according to Kenward and Roger (1997) for fixed terms in the dense part of the model.

dDF.na	The method to use to obtain substitute denominator degrees of freedom. when the numeric or algebraic methods produce an NA. If dDF.na = "none", no substitute denominator degrees of freedom are employed; if dDF.na = "residual", the residual degrees of freedom from asreml.obj$nedf are used; if dDF.na = "maximum", the maximum of those denDF that are available, excluding that for the Intercept, is used; if all denDF are NA, asreml.obj$nedf is used. If dDF.na = "supplied", a vector of values for the denominator degrees of freedom is to be supplied in dDF.values. Any other setting is ignored and a warning message produced. Generally, substituting these degrees of freedom is anticonservative in that it is likely that the degrees of freedom used will be too large.
dDF.values	A vector of values to be used when dDF.na = "supplied". Its values will be used when denDF in a test for a fixed effect is NA. This vector must be the same length as the number of fixed terms, including (Intercept) whose value could be NA.
trace	if TRUE then partial iteration details are displayed when ASReml-R functions are invoked; if FALSE then no output is displayed.
...	further arguments passed to asreml and to wald.asreml.

Value

A wald.tab: a 4-column data.frame containing a pseudo-anova table for the fixed terms produced by wald.asreml.

Author(s)

Chris Brien

See Also

[asrtests](#), [testranfix.asrtests](#)

Examples

```
## Not run:
  wald.tab <- recalcWaldTab(current.asrt,
                            dDF.na = "supplied",
                            dDF.values = c(NA,rep(c(330,346), c(4,3))))

## End(Not run)
```

---

redoErrorIntervals.alldiffs

Adds or replaces the error intervals stored in a prediction component of an alldiffs.object.

Description

Given an alldiffs object, adds or replaces error.intervals for its prediction component.

Usage

```
## S3 method for class 'alldiffs'
redoErrorIntervals(alldiffs.obj, error.intervals = "Confidence",
                   alpha = 0.05, avsed.tolerance = 0.25,
                   meanLSD.type = "overall", LSDby = NULL, ...)
```

Arguments

alldiffs.obj   An object of S3-class alldiffs (see `as.alldiffs`).

error.intervals

A `character` string indicating the type of error interval, if any, to calculate in order to indicate uncertainty in the results. Possible values are "none", "StandardError", "Confidence" and "halfLeastSignificant". The default is for confidence limits to be used. The "halfLeastSignificant" option results in half the Least Significant Difference (LSD) being added and subtracted to the predictions, the LSD being calculated using the square root of the mean of the variances of all or a subset of pairwise differences between the predictions. If meanLSD.type is set to overall, the avsed.tolerance is not NA and the range of the SEDs divided by the average of the SEDs exceeds avsed.tolerance then the error.intervals calculations and the plotting will revert to confidence intervals. Also, half LSDs cannot be used for backtansformed values and so confidence intervals will be used instead.

alpha   The significance level for an LSD to compare a pair of predictions.

avsed.tolerance

A `numeric` giving the value of the SED range, the range of the SEDs divided by the square root of the mean of the variances of all or a subset of the pairwise differences, that is considered reasonable in calculating error.intervals. It should be a value between 0 and 1. The following rules apply:

1. If avsed.tolerance is NA then mean LSDs of the type specified by meanLSD.type are calculated and used in error.intervals and plots.

2. Irrespective of the setting of meanLSD.type, if avsed.tolerance is not exceeded then the mean LSDs are used in error.intervals and plots.

3. If meanLSD.type is set to overall, avsed.tolerance is not NA, and avsed.tolerance is exceeded then error.intervals and plotting revert to confidence intervals.

4. If meanLSD.type is set to factor.combinations and avsed.tolerance is not exceeded for any factor combination then the half LSDs are used in error.intervals and plots; otherwise, error.intervals and plotting revert to confidence intervals.

5. If meanLSD.type is set to per.prediction and avsed.tolerance is not exceeded for any prediction then the half LSDs are used in error.intervals and plots; otherwise, error.intervals and plotting revert to confidence intervals.

meanLSD.type   A `character` string determining whether the mean LSD stored is (i) the overall mean, based on the square root of the mean of the variances of all pairwise variances, (ii) the mean for each factor.combination of the `factor`s specified by LSDby, which is based on the square root of the mean of the variances for all pairwise differences for each factor combination, or (iii) the per.prediction mean, based, for each prediction, on the square root of the mean of the variances for all pairwise differences involving that prediction. It also determines,

in conjunction with `avsed.tolerance`, which LSD will be used in calculating `error.intervals` and, hence, is used for plots.

LSDby   A [character](#) (vector) of factor names, being the names of the [factor](#)s for each of whose levels combinations a mean LSD, minLSD and max LSD is stored in the LSD component of the `alldiffs` object when `meanLSD.type` is `factor.combinatons`.

...   provision for passsing arguments to functions called internally - not used at present.

Value

An `alldiffs` object with predictions and their standard errors and, depending on the settings of the arguments, all pairwise differences between predictions, their standard errors and p-values and LSD statistics. If `power.transform` is not one, it will contain a `data.frame` with the backtransformed predictions. If `error.intervals` is not `"none"`, then the `predictions` component and, if present, the `backtransforms` component will contain columns for the lower and upper values of the limits for the interval. The names of these columns will consist of three parts separated by full stops: 1) the first part will be `lower` or `upper`; 2) the second part will be one of `Confidence`, `StandardError` or `halfLeastSignificant`; 3) the third component will be `limits`.

The name of the `response`, the `term`, the `classify` and `tdf`, as well as the degrees of freedom of the standard error, will be set as attributes to the object.

Author(s)

Chris Brien

See Also

[as.alldiffs](#), [print.alldiffs](#), [sort.alldiffs](#), [subset.alldiffs](#),
[allDifferences.data.frame](#), [recalcLSD.alldiffs](#), [predictPresent.asreml](#),
[plotPredictions.data.frame](#), `as.Date`, `predict.asreml`

Examples

```
## Not run:
data(WaterRunoff.dat)
asreml.options(keep.order = TRUE) #required for asreml4 only
current.asr <- asreml(fixed = pH ~ Benches + (Sources * (Type + Species)),
                      random = ~ Benches:MainPlots,
                      keep.order=TRUE, data= WaterRunoff.dat)
current.asrt <- asrtests(current.asr, NULL, NULL)
diffs <- predictPlus(classify = "Sources:Type",
                     asreml.obj = current.asr,
                     wald.tab = current.asrt$wald.tab,
                     present = c("Sources", "Type", "Species"))
diffs <- redoErrorIntervals.alldiffs(diffs, error.intervals = "halfLeastSignificant")

## End(Not run)
```

REMLRT.asreml   *Performs a REML ratio test to compare two models.*

Description

Extracts the REML log likelihood and the number of variance parameters from two `asreml` objects. It assumes that that the first `asreml` object corresponds to the null hypothesis and the second `asreml` object to the alternative hypothesis for the test being conducted. That is, the second `asreml` object is the result of fitting a model that is a reduced version of the model for the first object. In the case where the reduced model is obtained by setting positively-constrained variance parameters in the full model to zero, the `positive.zero` argument should be set to `TRUE` so that the p-value is computed using a mixture of chi-square distributions as described in Self and Liang (1987).

The function checks that the models do not differ in either their fixed or sparse models.

Usage

```
## S3 method for class 'asreml'
REMLRT(h0.asreml.obj, h1.asreml.obj,
       positive.zero = FALSE, bound.test.parameters = "none",
       DF = NULL, bound.exclusions = c("F","B","S","C"), ...)
```

Arguments

h0.asreml.obj   asreml object containing the fit under the model for the null hypothesis.

h1.asreml.obj   asreml object containing the fit under the model for the alternative hypothesis.

positive.zero   Indicates whether the hypothesized values for the variance components being tested are on the boundary of the parameter space. For example, this is true for positively-constrained variance components that, under the reduced model, are zero. This argument does not need to be set if `bound.test.parameters` is set.

bound.test.parameters

Indicates whether for the variance components being tested, at least some of the hypothesized values are on the boundary of the parameter space. The possibilities are "none", "onlybound" and "one-and-one". The default is "none", although if it is set to "none" and `positive.zero` is `TRUE` then `bound.test.parameters` is taken to be "onlybound". When `bound.test.parameters` is set to "one-and-one", it signifies that there are two parameters being tested, one of which is bound and the other is not. For example, the latter is true for testing a covariance and a positively-constrained variance component that, under the reduced model, are zero.

DF   A `numeric` giving the difference between the two models in the number of variance parameters whose estimates are not of the type specified in `bound.exclusions`. If `NULL` then this is determined from the information in `full.asreml.obj` and `reduced.asreml.obj`.

bound.exclusions

A `character` specifying one or more bound (constraint) codes that will result in a variance parameter being excluded from the count of estimated variance parameters. If set to `NULL` then none will be excluded.

...   Provision for passsing arguments to functions called internally - not used at present.

Value

A `data.frame` containing the log of the likelihood ratio, its degrees of freedom, its p-value and the number of bound parameters in each of the two models being compared.

Note

If `DF` is not `NULL`, the supplied value is used. Otherwise `DF` is determined from the information in `h1.asreml.obj` and `h0.asreml.obj`. In this case, the degrees of freedom for the test are computed as the difference between the two models in the number of variance parameters whose estimates do not have a code for bound specified in `bound.exclusions`.

If ASReml-R version 4 is being used then the codes specified in `bound.exclusions` are not restricted to a subset of the default codes, but a warning is issued if a code other than these is specified. For ASReml-R version 3, only a subset of the default codes are allowed: `F` (Fixed), `B` (Boundary), `C` (Constrained) and `S` (Singular).

The test statistic is calculated as $2(log(REML)_1 - log(REML)_0)$.

This procedure is only approriate when the null hypothesis is that (i) all parameters are on the boundary of the parameter space (ii) all parameters are in the interior of the parameter space, or (iii) there are two parameters, one of which is on the boundary and the other is not. Other cases have been discussed by Self and Liang (1987), but are not implemented here.

Author(s)

Chris Brien

References

Self, S.G., and Liang, K-Y. (1987) Asymptotic Properties of Maximum Likelihood Estimators and Likelihood Ratio Tests Under Nonstandard Conditions. *Journal of the American Statistical Association*, **82**, 605-10.

See Also

[infoCriteria.asreml](), [testranfix.asrtests]()

Examples

```
## Not run:
    REMLRT(ICV.max, ICV.red, bound.test.parameters = "onlybound")

## End(Not run)
```

---

reorderClassify.alldiffs

Reorders the components in an alldiffs object according to a new classify.

Description

All the components of an `alldiffs` object are re-ordered to be in standard order for the variables in a new classify, using [allDifferences.data.frame]().

Usage

```
## S3 method for class 'alldiffs'
reorderClassify(alldiffs.obj, newclassify,
                sortFactor = NULL, sortWithinVals = NULL,
                sortOrder = NULL, decreasing = FALSE, ...)
```

Arguments

alldiffs.obj   An object of S3-class alldiffs (see as.alldiffs).

newclassify   a character string giving the variables that define the margins of the multiway
                table that was predicted, but ordered so that the predictions are in the desired
                order when they are arranged in standard order for the newclassify. Multiway
                tables are specified by forming an interaction type term from the classifying
                variables, that is, separating the variable names with the : operator.

sortFactor   A character containing the name of the factor that indexes the set of pre-
                dicted values that determines the sorting of the components. If there is only one
                variable in the classify term then sortFactor can be NULL and the order is de-
                fined by the complete set of predicted values. If there is more than one variable
                in the classify term then sortFactor must be set. In this case the sortFactor
                is sorted for the predicted values within each combination of the values of the
                sortWithin variables: the classify variables, excluding the sortFactor. There
                should be only one predicted value for each unique value of sortFactor within
                each set defined by a combination of the values of the sortWithin variables.

sortWithinVals A list with a component named for each factor and numeric that is a classify
                variable for the predictions, excluding sortFactor. Each component should
                contain a single value that is a value of the variable. The combination of this
                set of values will be used to define a subset of the predicted values whose or-
                der will define the order of sortFactor to be used for all combinations of the
                sortWithinVals variables. If sortWithinVals is NULL then the first value of
                each sortWithin variable in predictions component is used to define sortWithinVals.
                If there is only one variable in the classify then sortWithinVals is ignored.

sortOrder   A character vector whose length is the same as the number of levels for
                sortFactor in the predictions component of the alldiffs object. It speci-
                fies the desired order of the levels in the reordered components of the alldiffs
                object. The argument sortWithinVals is ignored.

                The following creates a sortOrder vector levs for factor f based on the values
                in x: levs <- levels(f)[order(x)].

decreasing   A logical passed to order that detemines whether the order is for increasing
                or decreasing magnitude of the predicted values.

...   further arguments passed to allDifferences.data.frame.

Details

First, the components of the alldiffs object is arranged in standard order for the newclassify.
Then predictions are reordered according to the settings of sortFactor, sortWithinVals, sortOrder
and decreasing (see sort.alldiffs for details).

Value

The alldiffs object supplied with the following components, if present, sorted: predictions,
vcov, backtransforms, differences, p.differences and sed. Also, the sortFactor and sortOrder
attributes are set.

Author(s)

Chris Brien

See Also

[as.alldiffs](), [allDifferences.data.frame](), [print.alldiffs](), [sort.alldiffs](),
[redoErrorIntervals.alldiffs](), [recalcLSD.alldiffs](),
[predictPlus.asreml](), [predictPresent.asreml]()

Examples

```
## Not run:
data(Smarthouse.dat)
#Set up without any sorting
m1.asr <- asreml(y1 ~ Genotype*A*B,
                 random=~Replicate/Mainplot/Subplot,
                 data=Smarthouse.dat)
current.asrt <- asrtests(m1.asr)
current.asrt <- rmboundary(current.asrt)
m1.asr <- current.asrt$asreml.obj
diffs <- predictPlus(m1.asr, classify = "Genotype:A:B",
                     wald.tab = current.asrt$wald.tab,
                     error.intervals = "Stand", tables = "none")

#Re-order predictions so have all Genotypes together for each combination of A and B
diffs.reord <- reorderClassify(diffs, newclassify = "A:B:Genotype")

## End(Not run)
```

---

reparamSigDevn.asrtests

Reparamterizes each random (deviations) term involving devn.fac *to a fixed term and ensures that the same term, with* trend.num *replacing* devn.fac*, is included if any other term with* trend.num *is included in* terms.

Description

This function reparamterizes each random (deviations) term involving devn.fac to a fixed term and ensures that the same term with trend.num replacing devn.fac is included if any other term with trend.num is included in terms. It also ansures that any term with spl{trend.num} replacing devn.fac in a term being reparameterized is removed from the model.

Usage

```
## S3 method for class 'asrtests'
reparamSigDevn(asrtests.obj,terms = NULL,
               trend.num = NULL, devn.fac = NULL,
               allow.unconverged = TRUE, checkboundaryonly = FALSE,
               denDF = "numeric", trace = FALSE, update = TRUE,
               set.terms = NULL, ignore.suffices = TRUE,
               bounds = "P", initial.values = NA,...)
```

Arguments

asrtests.obj	an [asrtests](#) object for a fitted model that is a list containing an asreml object, a wald.tab data.frame with 4 columns, and a data.frame with 5 columns that records any previous changes and tests in the fitted model.
terms	a character string vector giving the terms that are to be reparameterized.
trend.num	A character string giving the name of the numeric covariate that corresponds to devn.fac and is potentially included in terms in the fitted model.
devn.fac	A character string giving the name of the factor that corresponds to trend.num and is included in terms in the fitted model.
allow.unconverged	
	A logical indicating whether to accept a new model even when it does not converge. Initially all changes are made with allow.unconverged set to TRUE. If allow.unconverged has been set to FALSE in the call and the final fit does not converge, an attempt is made to achieve convergence by removing any boundary terms. If this is unsuccessful, the supplied asrtsts object is returned.
checkboundaryonly	
	if TRUE then boundary and singular terms are not removed by [rmboundary.asrtests](#); a warning is issued instead.
denDF	Specifies the enthod to use in computing approximate denominator degrees of freedom when wald.asreml is called. Can be none to suppress the computations, numeric for numerical methods, algebraic for algebraic methods or default, the default, to autommatically choose numeric or algebraic computations depending on problem size. The denominator degrees of freedom are calculated according to Kenward and Roger (1997) for fixed terms in the dense part of the model.
trace	if TRUE then partial iteration details are displayed when ASReml-R functions are invoked; if FALSE then no output is displayed.
update	if TRUE then update.asreml is called in removing and adding terms to the model. In doing this the arguments R.param and G.param are set to those in the asreml object stored in the supplied asrtests.obj so that the values from the previous model are used as starting values. If FALSE then calls are made to asreml in which the only changes from the previous call are (i) that the models are updated and (ii) modifications specified via ... are made.
set.terms	a character vector specifying the terms that are to have bounds and/or initial values set prior to fitting.
ignore.suffices	
	a logical vector specifying whether the suffices of the asreml-assigned names of the variance terms (i.e. the information to the right of an "!", other than "R!") is to be ignored in matching elements of terms. If TRUE for an element of terms, the suffices are stripped from the asreml-assigned names. If FALSE for an element of terms, the element must exactly match an asreml-assigned name for a variance term. This vector must be of length one or the same length as terms. If it is of length one then the same action is applied to the asreml-assigned suffices for all the terms in terms.
bounds	a [character](#) vector specifying the bounds to be applied to the terms specified in set.terms. This vector must be of length one or the same length as set.terms. If it is of length one then the same constraint is applied to all the terms in set.terms. If any of the bounds are equal to NA then they are left unchanged for those terms.

initial.values   a character vector specifying the initial values for the terms specified in terms. This vector must be of length one or the same length as terms. If it is of length one then the same initial value is applied to all the terms in terms. If any of the initial.values are equal to NA then they are left unchanged for those terms.

...   further arguments passed to asreml via changeTerms.asrtests and asrtests.

Value

An asrtests object, which is a list containing:

1. asreml.obj: an asreml object containing the fit of the model after all boundary and singular terms have been removed;

2. wald.tab: a 4-column data.frame containing a pseudo-anova table for the fixed terms produced by wald.asreml;

3. test.summary: a data.frame with columns term, DF, denDF, p and action. It contains a row for each term that is dropped, added or tested or a note that several terms have been added or removed.

Author(s)

Chris Brien

See Also

asrtests, changeTerms.asrtests, testranfix.asrtests, testresidual.asrtests, newfit.asreml, chooseModel.asrtests

Examples

```
## Not run:
  data(WaterRunoff.dat)
  asreml.options(keep.order = TRUE) #required for asreml4 only
  current.asr <- asreml(fixed = log.Turbidity ~ Benches + Sources + Type + Species +
                                Sources:Type + Sources:Species + Sources:Species:xDay +
                                Sources:Species:Date,
                        data = WaterRunoff.dat, keep.order = TRUE)
  current.asrt <- asrtests(current.asr, NULL, NULL)

  #Examine terms that describe just the interactions of Date and the treatment factors
  terms.treat <- c("Sources", "Type", "Species", "Sources:Type", "Sources:Species")
  date.terms <- sapply(terms.treat,
                       FUN=function(term){paste("Date:",term,sep="")},
                       simplify=TRUE)
  date.terms <- c("Date", date.terms)
  date.terms <- unname(date.terms)
  treat.marginality <-  matrix(c(1,0,0,0,0,0, 1,1,0,0,0,0,  1,0,1,0,0,0,
                                 1,0,1,1,0,0, 1,1,1,0,1,0, 1,1,1,1,1,1), nrow=6)
  rownames(treat.marginality) <- date.terms
  colnames(treat.marginality) <- date.terms
  choose <- chooseModel(current.asrt, treat.marginality, denDF="algebraic")
  current.asrt <- choose$asrtests.obj
  current.asr <- current.asrt$asreml.obj
  sig.date.terms <- choose$sig.terms

  #Remove all Date terms left in the fixed model
```

```
terms <- "(Date/(Sources * (Type + Species)))"
current.asrt <- changeTerms(current.asrt, dropFixed = terms)
#if there are significant date terms, reparameterize to xDays + spl(xDays) + Date
if (length(sig.date.terms) != 0)
{ #add lin + spl + devn for each to fixed and random models
  trend.date.terms <- sapply(sig.date.terms,
                             FUN=function(term){sub("Date","xDay",term)},
                             simplify=TRUE)
  trend.date.terms <- paste(trend.date.terms,  collapse=" + ")
  current.asrt <- changeTerms(current.asrt, addFixed=trend.date.terms)
  trend.date.terms <- sapply(sig.date.terms,
                             FUN=function(term){sub("Date","spl(xDay)",term)},
                             simplify=TRUE)
  trend.date.terms <- c(trend.date.terms, sig.date.terms)
  trend.date.terms <- paste(trend.date.terms,  collapse=" + ")
  current.asrt <- changeTerms(current.asrt, addRandom = trend.date.terms)
  current.asrt <- rmboundary(current.asrt)
}

#Now test terms for sig date terms
spl.terms <- sapply(terms.treat,
                    FUN=function(term){paste("spl(xDay):",term,sep="")},
                    simplify=TRUE)
spl.terms <- c("spl(xDay)",spl.terms)
lin.terms <- sapply(terms.treat,
                    FUN=function(term){paste(term,":xDay",sep="")},
                    simplify=TRUE)
lin.terms <- c("xDay",lin.terms)
systematic.terms <- c(terms.treat, lin.terms, spl.terms, date.terms)
systematic.terms <- unname(systematic.terms)
treat.marginality <-  matrix(c(1,0,0,0,0,0, 1,1,0,0,0,0,  1,0,1,0,0,0,
                               1,0,1,1,0,0, 1,1,1,1,1,0, 1,1,1,1,1,1), nrow=6)
systematic.marginality <- kronecker(matrix(c(1,0,0,0, 1,1,0,0,
                                             1,1,1,0, 1,1,1,1), nrow=4),
                                    treat.marginality)
systematic.marginality <- systematic.marginality[-1, -1]
rownames(systematic.marginality) <- systematic.terms
colnames(systematic.marginality) <- systematic.terms
choose <- chooseModel(current.asrt, systematic.marginality,
                      denDF="algebraic", pos=TRUE)
current.asrt <- choose$asrtests.obj

#Check if any deviations are significant and, for those that are, go back to
#fixed dates
current.asrt <- reparamSigDevn(current.asrt, choose$sig.terms,
                               trend.num = "xDay", devn.fac = "Date",
                               denDF = "algebraic")

## End(Not run)
```

---

rmboundary.asrtests   *Removes any boundary or singular variance components from the fit*
                           stored in asrtests.obj *and records their removal in a data.frame.*

Description

Any terms specified in the random model that are estimated on the boundary or are singular and can be removed are removed from the fit stored in the asreml object stored in the asrtests object. Terms that specify multiple parameters in the random model cannot be removed (e.g. terms specified using the at function with more than one level of the factor) and terms in residual model are not removed. Terms that can be removed are selected for removal in the following order based on whether they involve: (i) a dev function, (ii) only factors, (iii) an spl function, (iv) a pol function and (v) a lin function or a variable that is an integer or a numeric. It should be noted that this order of removal presumes that random deviation terms are specified via the dev function rather than via a random factor. Once the earliest of the above classes with a boundary term is identified, a term within this class is selected for removal. For all classes, except for factor-only terms, the smallest term with the largest number of variables/factors is removed. Amongst factor-only terms, the smallest term with the smallest number of variables/factors is removed. After each variance component is removed, a row for it is added to the test.summary data.frame and the model refitted. If there are further boundary or singular terms, one is removed using the above strategy. This process continues until there are no further boundary or singular variance components that are removable. Other types of boundary or singular terms, which cannot be removed, are reported in warning messages.

Usage

```
## S3 method for class 'asrtests'
rmboundary(asrtests.obj, checkboundaryonly = FALSE,
           trace = FALSE, update = TRUE,
           set.terms = NULL, ignore.suffices = TRUE,
           bounds = "P", initial.values = NA, ...)
```

Arguments

asrtests.obj   an asrtests object for a fitted model that is a list containing an asreml object, a wald.tab data.frame with 4 columns, and a data.frame with 5 columns that records any previous changes and tests in the fitted model.

checkboundaryonly

        if TRUE then boundary and singular terms are not removed by rmboundary.asrtests; a warning is issued instead.

trace   if TRUE then partial iteration details are displayed when ASReml-R functions are invoked; if FALSE then no output is displayed.

update   if TRUE then update.asreml is called to fit the model with any boundary terms removed. In doing this the arguments R.param and G.param are set to those in the asreml object stored in asrtests.obj so that the values from the previous model are used as starting values. If FALSE then a call is made to asreml in which the only changes from the previous call are that (i) the terms for boundary variance components are removed from the models and (ii) modifications specified via ... are made.

set.terms   a character vector specifying the terms that are to have bounds and/or initial values set prior to fitting.

ignore.suffices

        a logical vector specifying whether the suffices of the asreml-assigned names of the variance terms (i.e. the information to the right of an "!", other than "R!") is to be ignored in matching elements of terms. If TRUE for an element of terms, the suffices are stripped from the asreml-assigned names. If FALSE

for an element of terms, the element must exactly match an asreml-assigned
name for a variance term. This vector must be of length one or the same length
as terms. If it is of length one then the same action is applied to the asreml-
assigned suffices for all the terms in terms.

bounds   a [character](#) vector specifying the bounds to be applied to the terms speci-
                      fied in set.terms. This vector must be of length one or the same length as
                      set.terms. If it is of length one then the same constraint is applied to all the
                      terms in set.terms. If any of the bounds are equal to NA then they are left
                      unchanged for those terms.

initial.values   a character vector specifying the initial values for the terms specified in terms.
                      This vector must be of length one or the same length as terms. If it is of length
                      one then the same initial value is applied to all the terms in terms. If any of the
                      initial.values are equal to NA then they are left unchanged for those terms.

...   further arguments passed to asreml.

Value

An [asrtests](#) object, which is a list containing:

1. asreml.obj: an asreml object containing the fit of the model after all boundary and singular
   terms have been removed;

2. wald.tab: a 4-column data.frame containing a pseudo-anova table for the fixed terms pro-
   duced by wald.asreml;

3. test.summary: a data.frame with columns term, DF, denDF, p and action. A row is added
   to it for each of the boundary terms removed, the row containing the name of the term, one for
   the DF, NA for the p-value and Boundary for the action.

Author(s)

Chris Brien

See Also

[asrtests](#), [changeTerms.asrtests](#), [testranfix.asrtests](#), [testresidual.asrtests](#),
[newfit.asreml](#), [reparamSigDevn.asrtests](#), [chooseModel.asrtests](#)

Examples

```
## Not run:
current.asrt <- rmboundary(current.asrt)

## End(Not run)
```

---

setvarianceterms.call   *allows the seting of bounds and initial values for terms in the* random
                         and residual *arguments of an* asreml *call, with the resulting call*
                         being evaluated.

Description

Takes an unevaluated `call` and evaluates the `call` after setting the bounds and initial values for the terms specified in `terms`. The elements of `terms` are matched with those generated by asreml and used, for example, in the `varcomp` component of a `summary.asreml` object. These names generally include descriptive suffices. To match an element of `terms` that includes such a suffix, set `ignore.suffices` to FALSE so that a literal match between the element and the assigned names is sought.

Usage

```
## S3 method for class 'call'
setvarianceterms(call, terms, ignore.suffices = TRUE,
                 bounds = "P", initial.values = NA, ...)
```

Arguments

`call`	an unevaluated `call` to `asreml`. One way to create such a `call` is to use the `call` function with its `name` argument set to `"asreml"`. Another is to obtain it from the `call` component of an asreml object (e.g. call <- asreml.obj$call).
`terms`	a character vector specifying the terms that are to have bounds and/or initial values specified.
`ignore.suffices`	a logical vector specifying whether the suffices of the `asreml`-assigned names of the variance terms (i.e. the information to the right of an "!", other than "R!") is to be ignored in matching elements of `terms`. If TRUE for an element of `terms`, the suffices are stripped from the `asreml`-assigned names. If FALSE for an element of `terms`, the element must exactly match an `asreml`-assigned name for a variance term. This vector must be of length one or the same length as `terms`. If it is of length one then the same action is applied to the `asreml`-assigned suffices for all the terms in `terms`.
`bounds`	a [character](#) vector specifying the bounds to be applied to the terms specified in `set.terms`. This vector must be of length one or the same length as `set.terms`. If it is of length one then the same constraint is applied to all the terms in `set.terms`. If any of the bounds are equal to NA then they are left unchanged for those terms.
	The codes used by ASReml are:
	• B - fixed at a boundary;
	• F - fixed by the user;
	• P - positive definite;
	• C - Constrained by user;
	• U - unbounded.
`initial.values`	a character vector specifying the initial values for the terms specified in `terms`. This vector must be of length one or the same length as `terms`. If it is of length one then the same initial value is applied to all the terms in `terms`. If any of the initial.values are equal to NA then they are left unchanged for those terms.
`...`	additional arguments to be added to the call, or arguments in the call with changed values.

Value

An `asreml` object.

Author(s)

Chris Brien

References

Butler, D. G., Cullis, B. R., Gilmour, A. R., Gogel, B. J. and Thompson, R. (2017). *ASReml-R User Guide Version 4*. VSN International Ltd, <http://www.vsni.co.uk/software/asreml/>.

See Also

`update.asreml`

Examples

```
## Not run:
   m1.call <- call("asreml",
                   fixed = Height ~ (Block + Irrig)*csDay.num,
                   random= ~ spl(csDay.num)/(Irrig+Block)
                             + dev(csDay.num)
                             + str(~Block:Plot/csDay.num, ~us(2):id(20))
                             + Block:Plot:spl(csDay.num),
                   data=quote(dat)) ##use quote to stop evaluation of dat here
   terms <- c("Block:Plot+Block:Plot:csDay.num!us(2).2:1", "R!variance")
   m1.asreml <- setvarianceterms(m1.call, terms, bounds=c("U","P"),
                                 initial=c(NA,3), ignore.suffices=c(FALSE,TRUE))
   summary(m1.asreml)

## End(Not run)
```

---

simulate.asreml	*Produce sets of simulated data from a multivariate normal distribtion and save quantites related to the simulated data*

Description

Produce in parallel sets of simulated data corresponding to an `asreml` model, along with its fitted values and residuals. A variance matrix V, corresponding to the `random` and `residual` models must be supplied. What to save is specified by the `which` argument.

Usage

```
## S3 method for class 'asreml'
simulate(object, nsim=100, seed = NULL, means=NULL, V, tolerance = 1E-10,
         update = TRUE, trace = FALSE, which="data", units = "ignore",
         ncores = detectCores(), ...)
```

Arguments

object	An `asreml` object from a call to `asreml` in which the `data` argument has been set.
means	The `vector` of means to be used in generating simulated data sets. If it is `NULL`, the fitted values based on `object` are used. It must be the same length as the response variable for `object`.

V	The fitted variance `matrix`, i.e. having the pattern and values that conform to the model fit stored in the supplied `object`.
nsim	The number of data sets to be simulated.
seed	A single value, interpreted as an integer, that specifies the starting value of the random number generator. The "L'Ecuyer-CMRG" random generator is used and `nextRNGStream` is used to seed each core from the original seed.
tolerance	The value such that eigenvalues less than it are considered to be zero.
update	if `TRUE` then the arguments `R.param` and `G.param` are set to those in the `asreml` object supplied in `object` so that the values from the original model are used as starting values. If `FALSE` then calls are made to `asreml` in which the only changes from the previous call are (i) the model is fitted to simulated data and (ii) modifications specified via ... are made, except that changes cannot be made to any of the models.
trace	if `TRUE` then partial iteration details are displayed when ASReml-R functions are invoked; if `FALSE` then no output is displayed.
which	The quantites from the simulated data set to be stored. Any combination of `"response"`, `"residuals"` and `"fitted"`, or `"all"`. If `residuals` and/or `fitted` is specified, those for the analysis stored in `object` will be added to the `data.frame` nominated in the `data` argument of `object` and the modified `data.frame` added as a component named `data` in the `list` that is the `value` returned by the `function`.
units	A `character` indicating whether the BLUPs for `units` are added to the residuals when this reserved factor is included in the `random` model. Possible values are `addtoresiduals` and `ignore`.
ncores	A `numeric` specifying the number of cores to use in doing the simulations.
...	Other arguments that are passed down to the function asreml. Changes to the models are not allowed. Other changes are dangerous and generally should be avoided.

Details

Generate `nsim` sets of data and analyse them using `asreml` using the model in `object`, performing the generation and analysis of several sets in parallel. Note, if the analysis for a data set does not converge in `maxiter` iterations, it is discarded and a replacement data set generated. The value of `maxiter` can be specified in the call to `simulate.asreml`. The fitted values and residuals are extracted as required. If `aom = TRUE` when the simulated data are analysed, standardised conditional residuals are stored. If `which` includes `residuals` or `fitted`, the specified quantities for the observed data are added to the `data.frame` on which the fit in `object` is based.

Value

A `list` with the following components whose presence depends on the setting of `which`:

1. **observed:** present if `which` includes `residuals` or `fitted`, in which case it will be the `data.frame` on which the fit in `object` is based, with `residuals` and/or `fitted`.

2. **data:** present if `which` includes `data`, a `data.frame` containing the simulated data sets.

3. **fitted:** present if `which` includes `fitted`, a `data.frame` containing the fitted values from the analyses of the simulated data sets.

4. **residuals:** present if `which` includes `residuals`, a `data.frame` containing the residuals from the analyses of the simulated data sets.

Author(s)

Chris Brien

See Also

asreml, `variofaces.asreml`, `plotVariofaces.data.frame`, `set.seed`.

Examples

```
## Not run:
data(Wheat.dat)
current.asr <- asreml(yield ~ Rep + WithinColPairs + Variety,
                      random = ~ Row + Column + units,
                      residual = ~ ar1(Row):ar1(Column),
                      data=Wheat.dat)
current.asrt <- asrtests(current.asr, NULL, NULL)
current.asrt <- rmboundary.asrtests(current.asrt)
# Form variance matrix based on estimated variance parameters
s2 <- current.asr$sigma2
gamma.Row <- current.asr$gammas[1]
gamma.unit <- current.asr$gammas[2]
rho.r <- current.asr$gammas[4]
rho.c <- current.asr$gammas[5]
row.ar1 <- mat.ar1(order=10, rho=rho.r)
col.ar1 <- mat.ar1(order=15, rho=rho.c)
V <- gamma.Row * fac.sumop(Wheat.dat$Row) +
     gamma.unit * diag(1, nrow=150, ncol=150) +
     mat.dirprod(col.ar1, row.ar1)
V <- s2*V

#Produce residuals from 100 simulated data sets
resid <- simulate(current.asr, V=V, which="residuals", ncores = 2)

## End(Not run)
```

---

sort.alldiffs	*Sorts the components in an alldiffs object according to the predicted values associated with a factor.*

Description

Sorts the rows of the components in an alldiffs object (see `as.alldiffs`) that are data.frames and the rows and columns of those that are matrices according to the predicted values in the predictions component. These predicted values are generally obtained using predict.asreml by specifying a classify term comprised of one or more variables. Generally, the values associated with one variable are sorted in parallel within each combination of values of the other variables. When there is more than one variable in the classify term, the sorting is controlled using one or more of sortFactor, sortWithinVals and sortOrder. If there is only one variable in the classify then all components are sorted according to the order of the complete set of predictions.

Note that reordering the classify variables in the alldiffs object and changing the order of the rows and columns of the components so that they are in standard order for the new variable order can be achieved using either `reorderClassify.alldiffs` or `allDifferences.data.frame`.

Usage

```
## S3 method for class 'alldiffs'
sort(x, decreasing = FALSE, classify = NULL,
     sortFactor = NULL, sortWithinVals = NULL, sortOrder = NULL, ...)
```

Arguments

x	An object of S3-class alldiffs.
decreasing	A [logical](#) passed to order that detemines whether the order is for increasing or decreasing magnitude of the predicted values.
classify	a [character](#) string giving the variables that define the margins of the multiway table that was predicted. Multiway tables are specified by forming an interaction type term from the classifying variables, that is, separating the variable names with the : operator. If NULL, it will be obtained from the classify attribute of the [as.alldiffs](#) object supplied through x.
sortFactor	A [character](#) containing the name of the factor that indexes the set of predicted values that determines the sorting of the components. If there is only one variable in the classify term then sortFactor can be NULL and the order is defined by the complete set of predicted values. If there is more than one variable in the classify term then sortFactor must be set. In this case the sortFactor is sorted for the predicted values within each combination of the values of the sortWithin variables: the classify variables, excluding the sortFactor. There should be only one predicted value for each unique value of sortFactor within each set defined by a combination of the values of the sortWithin variables.
sortWithinVals	A [list](#) with a component named for each factor and numeric that is a classify variable for the predictions, excluding sortFactor. Each component should contain a single value that is a value of the variable. The combination of this set of values will be used to define a subset of the predicted values whose order will define the order of sortFactor to be used for all combinations of the sortWithinVals variables. If sortWithinVals is NULL then the first value of each sortWithin variable in predictions component is used to define sortWithinVals. If there is only one variable in the classify then sortWithinVals is ignored.
sortOrder	A [character](#) vector whose length is the same as the number of levels for sortFactor in the predictions component of the alldiffs object. It specifies the desired order of the levels in the reordered components of the alldiffs object. The argument sortWithinVals is ignored. The following creates a sortOrder vector levs for factor f based on the values in x: levs <- levels(f)[order(x)].
...	further arguments passed to or from other methods. Not used at present.

Details

The basic technique is to change the order of the levels of the sortFactor within the predictions and, if present, backtransforms components so that they are ordered for a subset of predicted values, one for each levels of the sortFactor. When the classify term consists of more than one variable then a subset of one combination of the values of variables other than the sortFactor, the sortWithin set, must be chosen for determining the order of the sortFactor levels. Then the sorting of the rows (and columns) will be in parallel within each combination of the values of sortWithin variables: the classify term, excluding the sortFactor.

Value

The `alldiffs` object supplied with the following components, if present, sorted: `predictions`,
`vcov`, `backtransforms`, `differences`, `p.differences` and `sed`. Also, the `sortFactor` and `sortOrder`
attributes are set.

Author(s)

Chris Brien

See Also

`as.alldiffs`, `allDifferences.data.frame`, `print.alldiffs`,
`reorderClassify.alldiffs`, `redoErrorIntervals.alldiffs`, `recalcLSD.alldiffs`,
`predictPlus.asreml`, `predictPresent.asreml`

Examples

```
## Not run:
data(Smarthouse.dat)
#Set up without any sorting
m1.asr <- asreml(y1 ~ Genotype*A*B,
                 random=~Replicate/Mainplot/Subplot,
                 data=Smarthouse.dat)
current.asrt <- asrtests(m1.asr)
current.asrt <- rmboundary(current.asrt)
m1.asr <- current.asrt$asreml.obj
diffs <- predictPlus(m1.asr, classify = "Genotype:A:B",
                     wald.tab = current.asrt$wald.tab,
                     error.intervals = "Stand", tables = "none")

#Use sort.alldiffs and save order for use with other response variables
diffs.sort <- sort(diffs, sortFactor = "Genotype")
sort.order <- attr(diffs.sort, which = "sortOrder")

#Use sort.alldiffs with y1 sortOrder to sort y2 alldiffs object
m2.asr <- asreml(y2 ~ Genotype*A*B,
                 random=~Replicate/Mainplot/Subplot,
                 data=Smarthouse.dat)
current.asrt <- asrtests(m2.asr)
#Get predictions, sorted according to sort.order
diffs2.sort <- predictPlus(m2.asr, classify = "Genotype:A:B",
                           wald.tab = current.asrt$wald.tab,
                           error.intervals = "Stand", tables = "none",
                           sortFactor = "Genotype", sortOrder = sort.order)

## End(Not run)
```

---

subset.alldiffs   *Subsets the components in an alldiffs object according to the supplied*
                                        condition.

Description

Subsets each of the components of an `alldiffs` object. The subset is determined by applying the condition to the `prediction` component to determine which of its rows are to be included in the subset. Then, if present, this subset is applied to the rows of `backtransforms` and to the rows and columns of `differences`, `p.differences` and `sed` components. In addition, if `sed` is present, `recalcLSD.alldiffs` is called to recalculate the values in the `LSD` component, with any arguments supplied via the `...` argument passed ot it.

Usage

```
## S3 method for class 'alldiffs'
subset(x, subset, rmClassifyVars = NULL, ...)
```

Arguments

x	An object of S3-class `alldiffs`.
subset	A `logical` that detemines rows of the `predictions` component of x to be included in the subset.
rmClassifyVars	A `character` that contains the names of the variables in the `classify` attribute of x that are to be removed from the predictions `data.frame` and the names of the dimensions of the other components of x. It is advisable that this is only done when the combinations of the `rmClassifyVars` have only a single unique value; otherwise, the predictions and the rows an columns of the other components of x will not be unambiguously identified.
...	further arguments passed to `recalcLSD.alldiffs`.

Value

An `alldiffs` object with the following components of the supplied `alldiffs` object subsetted, if present in the original object: `predictions`, `vcov`, `backtransforms`, `differences`, `p.differences` and `sed`. In addition, if `sed` is present, the `LSD` component will be recalculated.

Author(s)

Chris Brien

See Also

`as.alldiffs`, `allDifferences.data.frame`, `print.alldiffs`, `sort.alldiffs`, `redoErrorIntervals.alldiffs`, `recalcLSD.alldiffs`, `predictPlus.asreml`, `predictPresent.asreml`

Examples

```
## Not run:
data(WaterRunoff.dat)
#Run analysis and produce alldiffs object
asreml.options(keep.order = TRUE) #required for asreml4 only
current.asr <- asreml(fixed = pH ~ Benches + (Sources * (Type + Species)),
                      random = ~ Benches:MainPlots,
                      keep.order=TRUE, data= WaterRunoff.dat)
current.asrt <- asrtests(current.asr, NULL, NULL)
diffs <- predictPlus.asreml(classify = "Sources:Type",
```

```
                            asreml.obj = current.asr, tables = "none",
                            wald.tab = current.asrt$wald.tab,
                            present = c("Type","Species","Sources"))


#Use subset.alldiffs to select a subset of the alldiffs object
diffs.subs <- subset(diffs,
                     subset = grepl("R", Sources, fixed = TRUE) &
                             Type %in% c("Control","Medicinal"))


## End(Not run)
```

---

testranfix.asrtests	*Tests for a single fixed or random term in model fitted using* asreml *and records the result in a data.frame.*

Description

Tests for a single term, using a REML LRT for a random term or based on Wald statistics for a fixed term. The term must be in the fitted model. A random term is removed from the model fit and a REML likelihood ratio test is performed using REMLRT.asreml. It compares the fit of the model in asreml.obj and the newly fitted model without the term. If the newly fitted model is retained, any boundary terms are then removed using rmboundary.asrtests. For a fixed term, the probability of the Wald statistics is extracted from the pseudo-anova table produced by wald.asreml. If this is available in the asrtests object, it is used; otherwise wald.asreml is called to add it to the asrtests object. Whether nonsignificant terms are dropped is controlled by drop.ran.ns for random terms and drop.fix.ns for fixed terms. A row is added to the test.summary data.frame for the term that is tested.

Usage

```
## S3 method for class 'asrtests'
testranfix(asrtests.obj, term=NULL, alpha = 0.05,
           allow.unconverged = TRUE, checkboundaryonly = FALSE,
           drop.ran.ns = TRUE, positive.zero = FALSE,
           bound.test.parameters = "none",
           bound.exclusions = c("F","B","S","C"), REMLDF = NULL,
           drop.fix.ns = FALSE, denDF="numeric", dDF.na = "none",
           dDF.values = NULL, trace = FALSE, update = TRUE,
           set.terms = NULL, ignore.suffices = TRUE,
           bounds = "P", initial.values = NA, ...)
```

Arguments

asrtests.obj	An asrtests object for a fitted model that is a list containing an asreml object, a wald.tab data.frame with 4 columns, and a data.frame with 5 columns that records any previous changes and tests in the fitted model.
term	A single model term that is valid in asreml, stored as a character.
alpha	The significance level for the test.

allow.unconverged

> A `logical` indicating whether to accept a new model even when it does not converge. If FALSE, it will be checked whether convergence can be achieved with the removal of any boundary random terms; random terms will be retested if terms are removed. Also, if FALSE and the fit of the new model has converged, but that of the old model has not, the new model will be accepted.

checkboundaryonly

> if TRUE then boundary and singular terms are not removed by [rmboundary.asrtests](#); a warning is issued instead.

drop.ran.ns

> A `logical` indicating whether to drop a random term from the model when it is nonsignificant.

positive.zero

> Indicates whether the hypothesized values for the variance components being tested are on the boundary of the parameter space. For example, this is true for positively-constrained variance components that, under the reduced model, are zero. This argument does not need to be set if bound.test.parameters is set.

bound.test.parameters

> Indicates whether for the variance components being tested, at least some of the hypothesized values are on the boundary of the parameter space. The possibilities are "none", "onlybound" and "one-and-one". The default is "none", although if it is set to "none" and positive.zero is TRUE then bound.test.parameters is taken to be "onlybound". When bound.test.parameters is set to "one-and-one", it signifies that there are two parameters being tested, one of which is bound and the other is not. For example, the latter is true for testing a covariance and a positively-constrained variance component that, under the reduced model, are zero.

bound.exclusions

> A `character` specifying one or more bound (constraint) codes that will result in a variance parameter being excluded from the count of estimated variance parameters in using [REMLRT.asreml](#). If set to NULL then none will be excluded.

REMLDF

> A `numeric` giving the difference in the number of variance parameters whose estimates are not of the type specified in bound.exclusions for two models being compared in a REML ratio test using [REMLRT.asreml](#). If NULL then this is determined from the information in the asreml object for the two models.

drop.fix.ns

> a `logical` indicating whether to drop a fixed term from the model when it is nonsignificant

denDF

> Specifies the enthod to use in computing approximate denominator degrees of freedom when wald.asreml is called. Can be none to suppress the computations, numeric for numerical methods, algebraic for algebraic methods or default, the default, to autommatically choose numeric or algebraic computations depending on problem size. The denominator degrees of freedom are calculated according to Kenward and Roger (1997) for fixed terms in the dense part of the model.

dDF.na

> the method to use to obtain substitute denominator degrees of freedom. when the numeric or algebraic methods produce an NA. If dDF.na = "none", no substitute denominator degrees of freedom are employed; if dDF.na = "residual", the residual degrees of freedom from asreml.obj$nedf are used; if dDF.na = "maximum", the maximum of those denDF that are available, excluding that for the Intercept, is used; if all denDF are NA, asreml.obj$nedf is used. If dDF.na = "supplied", a vector of values for the denominator degrees of freedom is to be supplied in dDF.values. Any other setting is ignored and a warning message produced.

Generally, substituting these degrees of freedom is anticonservative in that it is likely that the degrees of freedom used will be too large.

dDF.values   A vector of values to be used when dDF.na = "supplied". Its values will be used when denDF in a test for a fixed effect is NA. This vector must be the same length as the number of fixed terms, including (Intercept) whose value could be NA.

trace   if TRUE then partial iteration details are displayed when ASReml-R functions are invoked; if FALSE then no output is displayed.

update   if TRUE then update.asreml is called to fit the model to be tested. In doing this the arguments R.param and G.param are set to those in the asreml object stored in asrtests.obj so that the values from the previous model are used as starting values. If FALSE then a call is made to asreml in which the only changes from the previous call are that (i) models are modifed for the supplied terms and (ii) modifications specified via ... are made.

set.terms   a character vector specifying the terms that are to have bounds and/or initial values set prior to fitting.

ignore.suffices
                 a logical vector specifying whether the suffices of the asreml-assigned names of the variance terms (i.e. the information to the right of an "!", other than "R!") is to be ignored in matching elements of terms. If TRUE for an element of terms, the suffices are stripped from the asreml-assigned names. If FALSE for an element of terms, the element must exactly match an asreml-assigned name for a variance term. This vector must be of length one or the same length as terms. If it is of length one then the same action is applied to the asreml-assigned suffices for all the terms in terms.

bounds   a [character](#) vector specifying the bounds to be applied to the terms specified in set.terms. This vector must be of length one or the same length as set.terms. If it is of length one then the same constraint is applied to all the terms in set.terms. If any of the bounds are equal to NA then they are left unchanged for those terms.

initial.values   a character vector specifying the initial values for the terms specified in terms. This vector must be of length one or the same length as terms. If it is of length one then the same initial value is applied to all the terms in terms. If any of the initial.values are equal to NA then they are left unchanged for those terms.

...   Further arguments passed to asreml, wald.asreml and [asrtests](#).

Value

An [asrtests](#) object, which is a list containing:

1. asreml.obj: an asreml object containing the fit after the term has been tested; it will be a new model if the term is nonsignificant and the appropriate argument out of drop.ran.ns and drop.fix.ns is TRUE;

2. wald.tab: a 4-column data.frame containing a pseudo-anova table for the fixed terms produced by wald.asreml;

3. test.summary: a data.frame with columns term, DF, denDF, p and action. A row is added to it for each term that is tested, the row containing the name of the term, the degrees of freedom (numerator DF for a Wald test and the number of extra paramters for a REML ratio tests), the p-value and a for the action taken. Possible codes are: Dropped, Retained, Significant,

Nonsignificant, Absent, Added, Removed and Boundary. If the changed model did not converge, Unconverged will be added to the code. Note that the logical asreml.obj$converge also reflects whether there is convergence.

If the `term` is not in the model, then the supplied asreml object will be returned. Also, `reml.test` will have the likelihood ratio and the p-value set to NA and the degrees of freedom to zero. Similarly, the row of `test.summary` for the `term` will have its name, DF set to NA, p-value set to NA, and action set to Absent.

Author(s)

Chris Brien

See Also

[asremlPlus-package](#), [asrtests](#), [chooseModel.asrtests](#), [REMLRT.asreml](#), [rmboundary.asrtests](#), [newfit.asreml](#), [reparamSigDevn.asrtests](#), [changeTerms.asrtests](#)

Examples

```
## Not run:
data(Wheat.dat)
current.asr <- asreml(yield ~ Rep + WithinColPairs + Variety,
                      random = ~ Row + Column + units,
                      residual = ~ ar1(Row):ar1(Column),
                      data=Wheat.dat)
current.asrt <- asrtests(current.asr, NULL, NULL)
current.asrt <- rmboundary(current.asrt)
# Test nugget term
current.asrt <- testranfix(current.asrt, "units", positive=TRUE)

## End(Not run)
```

---

testresidual.asrtests   *Fits a new residual formula, tests whether the change is significant and records the result in a data.frame.*

Description

Fits a new `residual formula` using asreml4 (a new rcov formula with version 3 of asreml) and tests whether the change is significant. If `simpler = FALSE` the model to be fitted must be more complex than the one whose fit has been stored in asrtests.obj. That is, the new model must have more parameters. However, if `simpler = TRUE` the model to be fitted must be simpler than the one whose fit has been stored in asrtests.obj in that it must have fewer parameters. Any boundary terms are removed using [rmboundary.asrtests](#), which may mean that the models are not nested. The test is a REML likelihood ratio test that is performed using [REMLRT.asreml](#), which is only valid if the models are nested. It compares the newly fitted model with the fit of the model in asrtest.obj. A row is added to the `test.summary` data.frame using the supplied label.

Usage

```
## S3 method for class 'asrtests'
testresidual(asrtests.obj, terms=NULL, label = "R model",
             simpler = FALSE, alpha = 0.05, allow.unconverged = TRUE,
             checkboundaryonly = FALSE, positive.zero = FALSE,
             bound.test.parameters = "none",
             bound.exclusions = c("F","B","S","C"), REMLDF = NULL,
             denDF="numeric", update = TRUE, trace = FALSE,
             set.terms = NULL, ignore.suffices = TRUE,
             bounds = "P", initial.values = NA, ...)
```

Arguments

asrtests.obj	an [asrtests](#) object for a fitted model that is a list containing an asreml object, a wald.tab data.frame with 4 columns, and a data.frame with 5 columns that records any previous changes and tests in the fitted model.
terms	a model for the residual argument in asreml4 (the rcov formula in older versions of asreml), stored as a character.
label	a character string to use as the label in test.summary and which indicates what is being tested.
simpler	a logical indicating whether the new model to be fitted is simpler than the already fitted model whose fit is stored in asrtests.obj.
alpha	the significance level for the test.
allow.unconverged	A logical indicating whether to accept a new model even when it does not converge. If FALSE and the fit of the new model does not converge, the supplied asreml object is returned. Also, if FALSE and the fit of the new model has converged, but that of the old model has not, the new model will be accepted.
checkboundaryonly	if TRUE then boundary and singular terms are not removed by [rmboundary.asrtests](#); a warning is issued instead.
positive.zero	Indicates whether the hypothesized values for the variance components being tested are on the boundary of the parameter space. For example, this is true for positively-constrained variance components that, under the reduced model, are zero. This argument does not need to be set if bound.test.parameters is set.
bound.test.parameters	Indicates whether for the variance components being tested, at least some of the hypothesized values are on the boundary of the parameter space. The possibilities are "none", "onlybound" and "one-and-one". The default is "none", although if it is set to "none" and positive.zero is TRUE then bound.test.parameters is taken to be "onlybound". When bound.test.parameters is set to "one-and-one", it signifies that there are two parameters being tested, one of which is bound and the other is not. For example, the latter is true for testing a covariance and a positively-constrained variance component that, under the reduced model, are zero.
bound.exclusions	A [character](#) specifying one or more bound (constraint) codes that will result in a variance parameter being excluded from the count of estimated variance parameters in using [REMLRT.asreml](#). If set to NULL then none will be excluded.

REMLDF   A numeric giving the difference in the number of variance parameters whose estimates are not of the type specified in bound.exclusions for two models being compared in a REML ratio test using REMLRT.asreml. If NULL then this is determined from the information in the asreml object for the two models.

denDF   Specifies the method to use in computing approximate denominator degrees of freedom when wald.asreml is called. Can be none to suppress the computations, numeric for numerical methods, algebraic for algebraic methods or default, the default, to autommatically choose numeric or algebraic computations depending on problem size. The denominator degrees of freedom are calculated according to Kenward and Roger (1997) for fixed terms in the dense part of the model.

update   if TRUE then update.asreml is called to fit the model with the residual (rcov) model supplied in terms. In doing this the arguments R.param and G.param are set to those in the asreml object stored in asrtests.obj so that the values from the previous model are used as starting values. If FALSE then a call is made to asreml in which the only changes from the previous call are that (i) residual (rcov) model is that specified in terms and (ii) modifications specified via ... are made.

trace   if TRUE then partial iteration details are displayed when ASReml-R functions are invoked; if FALSE then no output is displayed.

set.terms   a character vector specifying the terms that are to have bounds and/or initial values set prior to fitting.

ignore.suffices
                a logical vector specifying whether the suffices of the asreml-assigned names of the variance terms (i.e. the information to the right of an "!", other than "R!") is to be ignored in matching elements of terms. If TRUE for an element of terms, the suffices are stripped from the asreml-assigned names. If FALSE for an element of terms, the element must exactly match an asreml-assigned name for a variance term. This vector must be of length one or the same length as terms. If it is of length one then the same action is applied to the asreml-assigned suffices for all the terms in terms.

bounds   a character vector specifying the bounds to be applied to the terms specified in set.terms. This vector must be of length one or the same length as set.terms. If it is of length one then the same constraint is applied to all the terms in set.terms. If any of the bounds are equal to NA then they are left unchanged for those terms.

initial.values a character vector specifying the initial values for the terms specified in terms. This vector must be of length one or the same length as terms. If it is of length one then the same initial value is applied to all the terms in terms. If any of the initial.values are equal to NA then they are left unchanged for those terms.

...   Further arguments passed to asreml, wald.asreml and asrtests.

Value

An asrtests object, which is a list containing:

1. asreml.obj: an asreml object containing the fit after the term has been omitted from the model;

2. wald.tab: a 4-column data.frame containing a pseudo-anova table for the fixed terms produced by wald.asreml;

3. `test.summary`: a `data.frame` with columns `term`, `DF`, `denDF`, `p` and `action`. A row is added to it for each term that is dropped, added or tested or a note that several terms have been added or removed. A row contains the name of the term, the DF, the p-value and the action taken. Possible codes are: `Dropped`, `Retained`, `Swapped`, `Unswapped`, `Significant`, `Nonsignificant`, `Absent`, `Added`, `Removed` and `Boundary`. If the changed model did not converge, `Unconverged` will be added to the code. Note that the logical `asreml.obj$converge` also reflects whether there is convergence.

If the `term` is not in the model, then the supplied `asreml` object will be returned. Also, `reml.test` will have the likelihood ratio and the p-value set to NA and the degrees of freedom to zero. Similarly, the row of `test.summary` for the `term` will have its name, a p-value set to NA, and action set to Absent.

Author(s)

Chris Brien

See Also

asremlPlus-package, asrtests, changeTerms.asrtests,
chooseModel.asrtests, REMLRT.asreml, rmboundary.asrtests,
newfit.asreml, testswapran.asrtests, changeTerms.asrtests,
reparamSigDevn.asrtests

Examples

```
## Not run:
data(Wheat.dat)
current.asr <- asreml(yield ~ Rep + WithinColPairs + Variety,
                      random = ~ Row + Column + units,
                      residual = ~ ar1(Row):ar1(Column),
                      data=Wheat.dat)
current.asrt <- asrtests(current.asr, NULL, NULL)
current.asrt <- rmboundary(current.asrt)
# Test Row autocorrelation
current.asrt <- testresidual(current.asrt, "~ Row:ar1(Column)",
                             label="Row autocorrelation", simpler=TRUE)
print(current.asrt)

## End(Not run)
```

---

testswapran.asrtests   *Tests, using a REMLRT, the significance of the difference between current random model and one in which oldterms are dropped and newterms are added. The result is recorded in a data.frame.*

Description

Fits a new random model using `asreml` by removing `oldterms` and adding `newterms`. If `simpler = FALSE` the model to be fitted must be more complex than the one whose fit has been stored in `asrtests.obj`. That is, the new model must have more parameters. However, if `simpler = TRUE` the model to be fitted must be simpler than the one whose fit has been stored in `asrtests.obj` in that it must have fewer parameters. The test is a REML ratio test that is performed using REMLRT.asreml, which is

only valid if the models are nested. It compares the newly fitted model with the fit of the model in asrtest.obj. A row is added to the test.summary data.frame using the supplied label. If the newly fitted model is retained, any boundary terms are then removed using rmboundary.asrtests.

Usage

```
## S3 method for class 'asrtests'
testswapran(asrtests.obj, oldterms = NULL, newterms = NULL,
            label = "Swap in random model", simpler = FALSE,
            alpha = 0.05, allow.unconverged = TRUE, checkboundaryonly = FALSE,
            positive.zero = FALSE, bound.test.parameters = "none",
            bound.exclusions = c("F","B","S","C"), REMLDF = NULL,
            denDF="numeric", trace = FALSE, update = TRUE,
            set.terms = NULL, ignore.suffices = TRUE,
            bounds = "P", initial.values = NA, ...)
```

Arguments

asrtests.obj	An asrtests object for a fitted model that is a list containing an asreml object, a wald.tab data.frame with 4 columns, and a data.frame with 5 columns that records any previous changes and tests in the fitted model.
oldterms	Terms, stored as a character, that are to be removed from the random model using asreml.
newterms	Terms, stored as a character, that are to be added to the random model using asreml.
simpler	A logical indicating whether the new model to be fitted. after the changes made as a result of swapping oldterms for newterms, is simpler than the already fitted model whose fit is stored in asrtests.obj.
alpha	The significance level for the test.
allow.unconverged	
	A logical indicating whether to accept a new model even when it does not converge. If FALSE and the fit of the new model does not converge, the supplied asreml object is returned. Also, if FALSE and the fit of the new model has converged, but that of the old model has not, the new model will be accepted.
checkboundaryonly	
	if TRUE then boundary and singular terms are not removed by rmboundary.asrtests; a warning is issued instead.
label	A character string to use as the label in test.summary and which indicates what is being tested.
positive.zero	Indicates whether the hypothesized values for the variance components being tested are on the boundary of the parameter space. For example, this is true for positively-constrained variance components that, under the reduced model, are zero. This argument does not need to be set if bound.test.parameters is set.
bound.test.parameters	
	Indicates whether for the variance components being tested, at least some of the hypothesized values are on the boundary of the parameter space. The possibilities are "none", "onlybound" and "one-and-one". The default is "none", although if it is set to "none" and positive.zero is TRUE then bound.test.parameters is taken to be "onlybound". When bound.test.parameters is set to "one-and-one", it signifies that there are two parameters being tested, one of which is bound and the other is not. For example, the latter is true for testing a covariance and a

positively-constrained variance component that, under the reduced model, are
zero.

bound.exclusions

    A character specifying one or more bound (constraint) codes that will result
in a variance parameter being excluded from the count of estimated variance
parameters in using REMLRT.asreml. If set to NULL then none will be excluded.

REMLDF   A numeric giving the difference in the number of variance parameters whose
estimates are not of the type specified in bound.exclusions for two models
being compared in a REML ratio test using REMLRT.asreml. If NULL then this
is determined from the information in the asreml object for the two models.

denDF   Specifies the enthod to use in computing approximate denominator degrees of
freedom when wald.asreml is called. Can be none to suppress the compu-
tations, numeric for numerical methods, algebraic for algebraic methods or
default, the default, to autommatically choose numeric or algebraic compu-
tations depending on problem size. The denominator degrees of freedom are
calculated according to Kenward and Roger (1997) for fixed terms in the dense
part of the model.

trace   If TRUE then partial iteration details are displayed when ASReml-R functions
are invoked; if FALSE then no output is displayed.

update   If TRUE then update.asreml is called to change the model. In doing this the
arguments R.param and G.param are set to those in the asreml object stored in
asrtests.obj so that the values from the previous model are used as starting
values. If FALSE then a call is made to asreml in which the only changes from
the previous call are that (i) models are modifed for the supplied oldterms and
newterms, and (ii) modifications specified via ... are made.

set.terms   A character vector specifying the terms that are to have bounds and/or initial
values set prior to fitting.

ignore.suffices

    A logical vector specifying whether the suffices of the asreml-assigned names
of the variance terms (i.e. the information to the right of an "!", other than
"R!") is to be ignored in matching elements of terms. If TRUE for an element
of terms, the suffices are stripped from the asreml-assigned names. If FALSE
for an element of terms, the element must exactly match an asreml-assigned
name for a variance term. This vector must be of length one or the same length
as terms. If it is of length one then the same action is applied to the asreml-
assigned suffices for all the terms in terms.

bounds   a character vector specifying the bounds to be applied to the terms speci-
fied in set.terms. This vector must be of length one or the same length as
set.terms. If it is of length one then the same constraint is applied to all the
terms in set.terms. If any of the bounds are equal to NA then they are left
unchanged for those terms.

initial.values   A character vector specifying the initial values for the terms specified in terms.
This vector must be of length one or the same length as terms. If it is of length
one then the same initial value is applied to all the terms in terms. If any of the
initial.values are equal to NA then they are left unchanged for those terms.

...   Further arguments passed to asreml, wald.asreml and asrtests.

Value

An asrtests object, which is a list containing:

1. `asreml.obj`: an asreml object containing the fit after the `term` has been omitted from the model;

2. `wald.tab`: a 4-column `data.frame` containing a pseudo-anova table for the fixed terms produced by `wald.asreml`;

3. `test.summary`: a `data.frame` with columns `term`, `DF`, `denDF`, `p` and `action`. A row is added to it for each term that is dropped, added or tested or a note that several terms have been added or removed. A row contains the name of the term, the DF, the p-value and the action taken. Possible codes are: `Dropped`, `Retained`, `Swapped`, `Unswapped`, `Significant`, `Nonsignificant`, `Absent`, `Added`, `Removed` and `Boundary`. If the changed model did not converge, `Unconverged` will be added to the code. Note that the logical `asreml.obj$converge` also reflects whether there is convergence.

If the `term` is not in the model, then the supplied `asreml` object will be returned. Also, `reml.test` will have the likelihood ratio and the p-value set to `NA` and the degrees of freedom to zero. Similarly, the row of `test.summary` for the `term` will have its name, a p-value set to `NA`, and action set to `Absent`.

Author(s)

Chris Brien

See Also

[asrtests](#), [chooseModel.asrtests](#), [REMLRT.asreml](#), [rmboundary.asrtests](#), [newfit.asreml](#), [testresidual.asrtests](#), [changeTerms.asrtests](#), [reparamSigDevn.asrtests](#)

Examples

```
## Not run:
current.asrt <- testswapran(current.asrt, oldterms = "str(~ Cart/xDays, ~us(2):id(184))",
                            newterms = "Cart/xDays", pos = FALSE,
                            label = "Intercept/Slope correlation",
                            simpler = TRUE)
  print(current.asrt)

## End(Not run)
```

---

variofaces.asreml	*Plots empirical variogram faces, including envelopes, as described by Stefanova, Smith & Cullis (2009).*

Description

A function that produces a plot for each face of an empirical 2D `variogram` based on residuals produced after the fitting of a model using the function `asreml`. It also adds envelopes to the plot by simulating data sets in parallel from a multivariate normal distribution with expectation equal to the fitted values obtained from the fixed and spline terms and variance matrix equal to the fitted variance matrix (Stefanova, Smith & Cullis, 2009). The plot is controlled by the `residual` model, which must consist of two factors corresponding to the two physical dimensions underlying the data. It can also have a third term involving the `at` or `dsum` function that defines sections of the data, such as experiments in different environments. In this case, the two variogram faces are produced for each section.

Usage

```
## S3 method for class 'asreml'
variofaces(asreml.obj, means=NULL, V=NULL, nsim=100, seed = NULL,
           extra.matrix = NULL, ignore.terms = NULL, fixed.spline.terms = NULL,
           bound.exclusions = c("F","B","S","C"), tolerance=1E-10,
           units = "ignore", update = TRUE, trace = FALSE,
           graphics.device=NULL, ncores = detectCores(), ...)
```

Arguments

asreml.obj	An asreml object from a call to asreml in which the data argument has been set.
means	The vector of means to be used in generating simulated data sets. If it is NULL, the fitted values based on object are used. It must be the same length as the response variable for object.
V	The fitted variance matrix, i.e. having the appropriate pattern and values given the model fitted to the observed data and the estimates of the parameters obtained. If V is NULL then estimateV.asreml is called to obtain it from asreml.obj
nsim	The number of data sets to be simulated in obtaining the envelopes.
seed	A single value, interpreted as an integer, that specifies the starting value of the random number generator. The "L'Ecuyer-CMRG" random generator is used and nextRNGStream is used to seed each core from the original seed.
extra.matrix	A matrix of order equal to the number of observations that is to be added to the varaince matrix, the latter based on the information in asreml.obj. It is assumed that the sigma-parameterized values of the variance parameter estimates, such as is given in the varcomp component of summary.asreml, have been used in calculating extra.matrix; the values in the vparameters component of G.param and R.param may be either gamma- or sigma-parameterized. The argument extra.matrix can be used in conjunction with ignore.terms as a workaround to include components of the variance matrix for variance functions that have not been implemented in estimateV.
ignore.terms	A character giving terms from either the random or residual models that are to be ignored in that their contributions to the variance is not to be included in the estimated matrix. The term names are those given in the vparameters component of the asreml object or the varcomp component produced by summary.asreml, but only up to the first exclamation mark (!). This can be used in conjunction with estimateV.asreml as a workaround to include components of the variance matrix for variance functions that have not been implemented in estimateV.
fixed.spline.terms	
	A character vector giving one or mor spline terms in the random model that are regarded as fixed and so are to be ignored because they are not regarded as contributing to the variance. The term names are those given in the vparameters component of the asreml object or the varcomp component produced by summary.asreml, but only up to the first exclamation mark (!).
bound.exclusions	
	A character specifying one or more bound codes that will result in a variance parameter in the random model being excluded from contributing to the variance. If set to NULL then none will be excluded.
tolerance	The value such that eigenvalues less than it are considered to be zero.

units	A `character` indicating whether the BLUPs for `units` are added to the residuals when this reserved factor is included in the `random` model. Possible values are `addtoresiduals` and `ignore`. If standardized conditional residuals are plotted and the BLUPs for `units` are to be added then it is the standardized BLUPs that are added.
update	if `TRUE` then the arguments `R.param` and `G.param` are set to those in the `asreml` object supplied in `object` so that the values from the original model are used as starting values. If `FALSE` then calls are made to `asreml` in which the only changes from the previous call are (i) the model is fitted to simulated data and (ii) modifications specified via ... are made, except that changes cannot be made to any of the models.
trace	if `TRUE` then partial iteration details are displayed when ASReml-R functions are invoked; if `FALSE` then no output is displayed.
graphics.device	
	A `character` specifying a graphics device for plotting. The default is `graphics.device = NULL`, which will result in plots being produced on the current graphics device. Setting it to `"windows"`, for example, will result in a windows graphics device being opened.
ncores	A `numeric` specifying the number of cores to use in doing the simulations.
...	Other arguments that are passed down to the function `asreml`. Changes to the models are not allowed. Other changes are dangerous and generally should be avoided.

Details

The `residual` model is scanned to ensure that it involves only two factors not included in the `at` function, and to see if it has a third factor in an `at` function. If so, the faces of the 2D variogram, each based on one of the two non-at factors, are derived from the residuals in the supplied `asreml` object using `asreml.variogram`, this yielding the observed `variogram` faces. If `aom` was set to `TRUE` for the `asreml` object, the standardized consitional residuals are used. Then `nsim` data sets are generated by adding the `fitted.values`, extracted from the `asreml` object, to a vector of values randomly generated from a normal distribution with expectation zero and variance matrix `V`. Each data set is analyzed using the model in `object` and several sets are generated and analyzed in parallel. The variogram values for the faces are obtained using `asreml.variogram` stored. Note, if the analysis for a data set does not converge in `maxiter` iterations, it is discarded and a replacement data set generated. The value of `maxiter` can be specified in the call to `variofaces.asreml`. Plots are produced for each face and include the observed values and the 2.5%, 50% & 97.5% quantiles.

Value

A `list` with the following components:

1. **face1:** a `data.frame` containing the variogram values on which the plot for the first dimension is based.

2. **face2:** a `data.frame` containing the variogram values on which the plot for the second dimension is based.

Author(s)

Chris Brien

References

Stefanova, K. T., Smith, A. B. & Cullis, B. R. (2009) Enhanced diagnostics for the spatial analysis of field trials. *Journal of Agricultural, Biological, and Environmental Statistics*, **14**, 392–410.

See Also

asremlPlus-package, asreml, plotVariofaces.data.frame, simulate.asreml, set.seed.

Examples

```
## Not run:
data(Wheat.dat)
current.asr <- asreml(yield ~ Rep + WithinColPairs + Variety,
                      random = ~ Row + Column + units,
                      residual = ~ ar1(Row):ar1(Column),
                      data=Wheat.dat)
current.asrt <- asrtests(current.asr, NULL, NULL)
current.asrt <- rmboundary.asrtests(current.asrt)
# Form variance matrix based on estimated variance parameters
s2 <- current.asr$sigma2
gamma.Row <- current.asr$gammas[1]
gamma.unit <- current.asr$gammas[2]
rho.r <- current.asr$gammas[4]
rho.c <- current.asr$gammas[5]
row.ar1 <- mat.ar1(order=10, rho=rho.r)
col.ar1 <- mat.ar1(order=15, rho=rho.c)
V <- gamma.Row * fac.sumop(Wheat.dat$Row) +
     gamma.unit * diag(1, nrow=150, ncol=150) +
     mat.dirprod(col.ar1, row.ar1)
V <- s2*V

#Produce variogram faces plot (Stefanaova et al, 2009)
variofaces(current.asr, V=V, ncores = 2)

## End(Not run)
```

WaterRunoff.dat	*Data for an experiment to investigate the quality of water runoff over time*

Description

This data is from an experiment to investigate the quality of water runoff. However, it has been modified to hide the true identity of the Species and Sources. It is used to provide executable examples of the functions listed under **Examples**.

Usage

```
data(WaterRunoff.dat)
```

Format

A data.frame containing 440 observations of 13 variables.

Author(s)

Chris Brien

Source

Kazemi, F. (pers. comm.)

See Also

[chooseModel.asrtests](#), [reparamSigDevn.asrtests](#),
[plotPredictions.data.frame](#), [predictPlus.asreml](#), [predictPresent.asreml](#)

Wheat.dat   *Data for an experiment to investigate 25 varieties of wheat*

Description

The data appears in Gilmour et al. [1995] and is from a field experiment designed to compare the performance of 25 varieties of wheat. An analysis of it using asreml is presented by Butler et al. (2010, Sectoion 8.6), although they suggest that it is a barley experiment. It is used in [asremlPlus-package](#) as an executable example of the use of the asremlPlus to analyse a data set.

The experiment was conducted at Slate Hall Farm, UK, in 1976 and was designed as a balanced lattice square with 6 replicates laid out in a 10×15 rectangular grid. The columns in the data frame are: Rep, Row, Column, WithinColPairs, Variety, yield. The response variable is the grain yield.

Usage

```
data(Wheat.dat)
```

Format

A data.frame containing 150 observations of 6 variables.

Author(s)

Chris Brien

Source

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

Gilmour, A. R., et al. (1995) Average Information REML: An efficient algorithm for variance parameter estimation in linear mixed models. *Biometrics*, **51**, 1440-1450.

Index