Package 'asreml'

June 11, 2024

Title Fits Linear Mixed Models using REML

Version 4.2.0.332

Date 2023-4-01

Author The VSNi Team

Maintainer The VSNi Team <asreml@vsni.co.uk>

Description ASReml-R is a statistical package that fits linear mixed models using Residual Maximum Likelihood (REML). The features of ASReml-R include: 1) a flexible syntax for specifying variance models for the random effects, and 2) the REML routines use the Average Information (AI) algorithm, and sparse matrix methods for fitting the mixed model. These enable ASReml-R to fit many types of models and efficiently analyse large and complex data sets.

Depends R (>= 4.0.0), stats, utils, Matrix

Imports data.table (>= 1.14), ggplot2, grid, methods, jsonlite

Suggests lattice, grDevices

Encoding UTF-8

License file LICENSE

URL www.vsni.co.uk

LazyData true

LazyDataCompression bzip2

ByteCompile yes

RoxygenNote 7.3.1

Language en-GB

Contents

`adjust.asreml.object` *Adjusts an object to match the previous version of ASReml-R*

Description

A function to take an object from the current main fitting routine (asreml) and adjust it to more closely match an object from the main fitting routine of the previous version of ASReml-R.

Usage

```
adjust.asreml.object(fit)
```

Arguments

`fit`	The object returned from the main asreml fitting routine

Details

This is done here rather than during one of the lower level of code basis for convenience

---

`ainverse` *Calculate an inverse relationship matrix*

Description

Generates an inverse of a numerator relationship matrix in sparse triplet form from a pedigree data frame.

Usage

```
ainverse(
  pedigree,
  fgen = list(character(0), 0.01),
  gender = character(0),
  groups = 0,
  groupOffset = 0,
  selfing = NA,
  inBreed = NA,
  mgs = FALSE,
  mv = NULL,
  psort = FALSE,
  core_version = NULL
)
```

Arguments

pedigree	A data frame where the first three columns correspond to the identifiers for the individual, male parent and female parent, respectively. The row giving the pedigree of an individual must appear before any row where that individual appears as a parent. Founders or unknown parents use 0 (zero) or NA in the parental columns.
fgen	An optional list of length two where fgen[[1]] is a character string naming the column in pedigree that contains the level of selfing or the level of inbreeding of an individual. In pedigree[,fgen[[1]]], 0 indicates a simple cross, 1 indicates selfed once, 2 indicates selfed twice, etc. A value between 0 and 1 for a base individual is taken as its inbreeding value. If the pedigree has implicit individuals (they appear as parents but not as individuals), they will be assumed base non-inbred individuals unless their inbreeding level is set with fgen[[2]], where 0 < fgen[[2]] < 1 is the inbreeding level of such individuals.
gender	An optional character string naming the column of pedigree that codes for the gender of an individual. pedigree[,gender] is coerced to a factor and must only have two (arbitrary) levels, the first of which is taken to mean "male". An inverse relationship matrix is formed for the X chromosome as described by *Fernando and Grossman, 1990* for species where the male is XY and the female is XX.
groups	An integer scalar (g) indicating genetic groups in the pedigree. The first g lines of the pedigree identify the genetic groups (with zero in both the male and female parent columns). All other rows must specify one of the genetic groups as the male or female parent if the actual parent is unknown (default g=0).
groupOffset	A numeric scalar $e > 0$ added to the diagonal elements of **A^-1** pertaining to groups, shrinking the group effects by e. When a constant is added, no adjustment of the degrees of freedom is made for genetic groups. Set to -1 to add no offset but to suppress insertion of constraints where empty groups appear; the empty groups are then not counted in the degrees of freedom adjustment (default $e = 0$).
selfing	A numeric scalar (s) allowing for partial selfing when the third field of pedigree is unknown. It indicates that progeny from a cross where the male parent is unknown is assumed to be from selfing with probability s and from outcrossing with probability $(1-s)$. This is appropriate in some forestry tree breeding studies where seed collected from a tree may have been pollinated by the mother tree or pollinated by some other tree (*Dutkowski and Gilmour, 2001*). Do not use the selfing argument in conjunction with inBreed or mgs.
inBreed	A numeric scalar giving the inbreeding coefficient for *base* individuals. This argument generates the numerator relationship matrix for inbred lines. Each cross is assumed to be selfed several times to stabilize as an inbred line as is usual for cereal crops, for example, before being evaluated or crossed with another line. Since inbreeding is usually associated with strong selection, it is not obvious that a pedigree assumption of covariance of 0.5 between parent and offspring actually holds. The inBreed argument cannot be used in conjunction with selfing or mgs (default = NA).
mgs	If TRUE, the third identity in the pedigree is the male parent of the female parent (maternal grand-sire) rather than the female parent (default = FALSE).

mv	A character vector of missing value indicators; elements of pedigree that exactly match any of the members of mv are treated as missing. The values passed are used in addition to "0" and "NA" which are hardcoded as missing value indicators.
psort	If TRUE, the pedigree data frame is returned in founder order after any insertions and permutations (default = FALSE).
core_version	The version of the algorithmic core to use, not usually required.

Details

Uses the method of *Meuwissen and Luo, 1992* to compute the inverse relationship matrix directly from the pedigree.

Value

A three-column matrix with class ginv holding the lower triangle of the inverse relationship matrix in sparse form. The first two columns are the *row* and *column* indices, respectively, and the third column holds the inverse matrix element itself. Sort order is columns within rows, that is, the lower triangle row-wise. This matrix has attributes:

rowNames A character vector of identifiers for the rows of the matrix.

inbreeding A numeric vector containing the inbreeding coefficient for each individual, calculated as diag(A-I).

geneticGroups A numeric vector of length two containing the groups and groupOffset arguments.

logdet The log determinant.

References

G W Dutkowski, A R Gilmour (2001). "Modification of the Additive Relationship Matrix for Open Pollinated Trials." In *Developing the Eucalypt of the Future*, 71. Instituto Forestal Chile, Valdivia. R Fernando, M Grossman (1990). "Genetic Evaluation with Autosomal and X-Chromosomal Inheritance." *Theoretical and Applied Genetics*, **80**, 75-80. T H E Meuwissen, Z Luo (1992). "Computing Inbreeding Coefficients in Large Populations." *Genetics Selection Evolution*, **24**, 305-313.

Examples

```
## Not run:

# Simple pedigree
ped <- data.frame(
  me  = c(1, 2, 3, 4, 5, 6, 7, 8, 9, 10),
  dad = c(0, 0, 0, 1, 1, 2, 4, 5, 7, 9),
  mum = c(0, 0, 0, 1, 1, 2, 6, 6, 8, 9))
p.ai <- ainverse(ped)

# Known filial generation
pdfg <- data.frame(
```

```
  me   = c(1, 2, 3, 4, 5, 6, 7),
  dad  = c(0, 0, 1, 1, 1, 1, 1),
  mum  = c(0, 0, 0, 2, 2, 2, 2),
  fgen = c(NA, 0.8, 0.0, 2.0, 0.0, 2.0, 3.0))
pdfg.ai <- ainverse(pdfg, fgen = list("fgen", 0.4))
pdfg.mat <- sp2mat(pdfg.ai)
zapsmall(solve(pdfg.mat))
zapsmall(cbind(pdfg.a$inbreeding, diag(pdfg.mat)))

## End(Not run)
```

---

asreml *Mixed effects linear models*

Description

The library `asreml()` estimates variance components under a general linear mixed model by residual maximum likelihood (REML).

Usage

```
asreml(
  fixed = ~1,
  random = ~NULL,
  sparse = ~NULL,
  residual = ~NULL,
  G.param = list(),
  R.param = list(),
  na.action = na.method(),
  subset,
  weights,
  predict = predict.asreml(),
  vcm = vcm.lm(),
  vcc = matrix(0),
  family = asr_gaussian(),
  asmv = NULL,
  mbf = list(),
  group = list(),
  equate.levels = character(0),
  start.values = FALSE,
  knot.points = list(),
  pwr.points = list(),
  wald = list(),
  prune = list(),
  combine = list(),
  uid = list(),
  mef = list(),
```

```
  last = list(),
  model.frame = TRUE,
  data = sys.parent(),
  ...
)
```

Arguments

`fixed`	A formula object specifying the fixed terms in the model, with the response on the left of a ~ operator, and the terms, separated by + operators, on the right. If `data` is given, all names used in all formulae should appear in the data frame. A model with the intercept as the only fixed effect can be specified as `~ 1`; there must be at least one fixed effect specified. If the response (y) evaluates to a matrix then a factor `trait` with levels `dimnames(y)[[2]]` is added to the model frame, and must be explicitly included in the model formulae (default = `NULL`).
`random`	A formula object specifying the random effects in the model. This argument has the same general characteristics as `fixed`, but there can be no left side to the ~ operator. Variance structures imposed on random terms are specified using special model functions (default = `~NULL`).
`sparse`	A formula object, specifying the fixed effects for which the full variance-covariance matrix is not required. This argument has the same general characteristics as `fixed`, but there can be no left side to the ~ expression. Wald statistics are not available for sparse fixed terms in order to reduce the computing load (default = `~NULL`).
`residual`	A formula object specifying the residual model; any term specified on the left of the ~ expression is ignored. The default is `~ units`, where the reserved word `units` is defined as `seq(1,nrow(data))` and is automatically included in the model frame. Variance models for the residual component of the model can be specified using special model functions (default = `~NULL`). For single-section univariate models, the residual variance model determines the computational mode: If the residual variance model specifies a correlation structure (includes `id()`), then the model is fitted on the gamma scale, otherwise the model is fitted on the sigma scale. The default is `id(units)` if not explicitly specified.
`G.param`	Either: • A list object derived from the `random` formula, holding initial parameter estimates and boundary constraints for each term, or • A character string naming a comma-delimited file with a header line and three columns for the variance component name, initial value and constraint code, respectively. This file can be created using the `start.values = TRUE` argument; the internal list object is then generated from the contents of this file. On termination, `G.param` is updated with the final `random` component estimates.
`R.param`	Either: • A list object derived from the `random` formula, holding initial parameter estimates and boundary constraints, or

- A character string naming a comma-delimited file with a header line and three columns for the variance component name, initial value and constraint code, respectively. This file can be created using the `start.values = TRUE` argument; the internal list object is then generated from the contents of this file.

On termination, `R.param` is updated with the final `residual` component estimates.

`na.action` A call to `na.method()` specifying the action to be taken when missing values are encountered in the response (`y`) or explanatory variables (`x` for factors and/or variates). The function definition for `na.method` is:

```
function(y = c("include", "omit", "fail"),
 x = c("fail", "include", "omit"))
```

The default action is `"include"` (and estimate) missing values in the response, and raise an error (`"fail"` if there are missing values in any of the explanatory variables.

`subset` A logical vector identifying which subset of the rows of `data` should be used in the fit. All observations are included by default.

`weights` A character string or name identifying the column of `data` to use as weights in the fit.

`predict` A list object specifying the classifying factors and related options when forming predictions from the model. This list would normally be the value returned by a call to the method `predict` for `asreml` objects.

`vcm` A matrix defining relationships among variance parameters. The matrix has a row for each original variance parameter and a column for each new parameter. The default is the identity matrix, that is, no action. See function `vcm.lm` for further information and an example.

`vcc` Equality constraints between variance parameters; a two-column numeric matrix with a `dimnames` attribute. The first column defines the *grouping* structure of equated components, that is, components within an equality *group* are given the same numeric index, and the second column contains the scaling coefficients. The `dimnames()[[1]]` attribute must match the component names in the `asreml` parameter vector; see `start.values`.

The parameters are scaled relative to the first parameter in its group, so the scaling of the first parameter in each group is one.

For example, the following `vcc` matrix:

$$\begin{matrix} 1 & 1 \\ 2 & 1 \\ 2 & 2 \\ 3 & 1 \end{matrix}$$

is equivalent to the `vcm` matrix:

$$\begin{matrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 2 & 0 \end{matrix}$$

0 0 1

`family` A list of functions and expressions for defining the link and variance functions.
Supported families are: *gaussian*, *inverse Gaussian*, *binomial*, *negative binomial*, *poisson* and *Gamma*. Family objects are generated from the `asreml` family distributions which prefix the usual function names with `"asr_"`; for example `asr_gaussian()`, `asr_binomial()`, etc. In addition to the link argument, these functions take an additional `dispersion` argument and a `total` argument where relevant; for example:
`asr_binomial(dispersion = 1.0, total = counts)`.
The default for `asr_gaussian()` is `dispersion = NA`, which implies that `asreml` will estimate the dispersion parameter, otherwise the scale is fixed at the nominated value.

`asmv` A character string or name specifying the column in the data that identifies the traits in a multivariate analysis. If not `NULL`, `asmv` implies that the data for a multivariate analysis is set up as though it were for a univariate analysis with the response in a single variate (default = `NULL`).

`mbf` A named list specifying sets of covariates to be included with one or more `mbf()` model functions. Each component of the list must in turn contain components named `key` and `cov`, where `cov` is a character string naming the data frame holding the covariates, and `key` is a character vector of length two naming the columns in `data` and `cov`, respectively, used to match corresponding records in the two data frames. The default is an empty list.

`group` A named list where each component is a numeric vector specifying contiguous fields in `data` that are to be considered as a single term. The component names can then appear in the `asreml` model formulae using the `grp()` special function. The default is an empty list.

`equate.levels` A character vector of factor names whose levels are to be *equated*. For example, if factor `A` has levels *a, b, c, d* and factor `B` has levels *a, b, c, e*, the effect of `equate.levels(A, B)` is that both `A` and `B` have five levels, with `as.numeric(A)` = *1, 2, 3, 4* and `as.numeric(B)` = *1, 2, 3, 5*. This may be necessary if using the `and()` model function to overlay columns of the model's design matrix in forming a compound term. The default is a zero-length character vector.

`start.values` If `TRUE`, `asreml` exits prior to the fitting process and returns a list of length three: the `G.param` and `R.param` lists, and a data frame (containing variance parameter names, initial values and boundary constraints). Initial values or constraints can then be set in the list or data frame objects (default = `FALSE`).
If this is a character string, then a file of that name is created and the data frame object containing initial parameter values is written out in comma-separated form. This file can be edited externally and subsequently specified in the `G.param` or `R.param` arguments.

`knot.points` A named list where each component is a vector of user-supplied knot points for a particular spline term; the component name is the object of the `spl()` model function.

`pwr.points` A named list with each component containing a vector of distances to be used in a one-dimensional power model. The component names must correspond to the `object` arguments of the power function model terms.

wald — A named list with three components: denDF, ssType and Ftest.

- denDF: A character string from the options: "none", "numeric", "algebraic", and "default" specifying the calculation of approximate denominator degrees of freedom. The option "none" is to suppress the computations. Algebraic computations are not feasible in large analyses; use "default" to automatically choose numeric or algebraic computations depending on problem size. The denominator degrees of freedom are calculated according to *Kenward and Roger, 1997* for terms in the fixed model formula (default = "none").
- ssType: It can be "incremental" for incremental sum of squares or "conditional" for F-tests that respect both structural and intrinsic marginality (default = "incremental".
- Ftest: A one-sided formula of the form ~ test_term | background_terms specifying a conditional Wald test of the contribution of test_term conditional on those fixed terms listed in background_terms, and the those in the random and sparse model formulae.

prune — A named list with each component generated from a call to Subset(). The argument prune, in conjunction with Subset and the model function sbs(), forms a new factor from an existing one by selecting a subset of its levels. The function Subset is defined as:

```
function(f, x)
```

where f is the name of an existing factor and x is a character or numeric vector of levels to select. The name of the list component is the new factor that may appear in the model formulae as the argument to the sbs() model function. For example,

```
prune = list(A = Subset(Site, c(2, 3)))
```

creates a new factor A by selecting the second and third levels of Site, and would be included in the model as: sbs(A), for example by using idv(sbs(A)) as part of a random term. While the actions of prune can be duplicated outside asreml, sbs() is necessary if the asreml method predict() is to be used.

combine — A named list with each component generated from a call to Levels(). The argument combine, in conjunction with Levels and the model function gpf(), forms a new factor from an existing one by merging a subset of its levels. The function Levels is defined as:

```
function(f, x)
```

where f is the name of an existing factor and x is a vector of length length(levels(f)) defining the levels of f to merge. The name of the list component is the new factor that may appear in the model formulae as the argument to the gpf() model function. For example, if Site has levels "1", "2" and "3",

```
combine = list(A = Levels(Site, c("1", "2", "1")))
```

creates a new factor A with levels "1" and "2" by merging levels "1" and "3" of Site, and would be included in the model as gpf(A). While the actions of combine can be duplicated outside asreml, gpf() is necessary if the asreml method predict() is to be used.

uid — A named list with each component generated from a call to Units(). The argument uid, in conjunction with Units and the model function uni(), forms a new factor by selecting a subset of records from an existing one. The function Units is defined as:

function(f, n = 0)

where f is the name of an existing factor and n is a character or numeric scalar that determines which records are selected. The default, n = 0, forms a factor with a level for each record where f is non-zero (strictly, f != 0). Otherwise, a factor with a level for each record in data where f has the value n is formed. For example,

uid = list(A = Units(group, 1))

creates a new factor A with levels from row.names(data) where group = 1, and would be included in the model as: uni(A). While the actions of uid can be duplicated outside asreml, uni() is necessary if the asreml method predict() is to be used.

mef — A named list linking a relationship matrix (or its inverse) as specified in the vm() special function with the original matrix of *subject* x *regressor* (typically molecular marker) scores. If this is not an empty list mef flags the computation of the *regressor* (marker) effects from the *subject* effects. For example,

mef = list(MM = snp.mat)

links the relationship matrix MM to the original marker scores found in the file snp.mat.

The mef list would typically be set from a call to the asreml meff() method.

last — A named list restricting the order equations are solved in the sparse partition for the nominated model terms. Each component of the list is named by a model term and contains a scalar n specifying that the first n levels of the term will be solved after all others in the sparse set. It is intended for use when there are multiple fixed terms in the sparse equations so that asreml will be consistent in which effects are identified as singular. A maximum of three factor/level pairs can be specified.

model.frame — If TRUE, the model frame (a data.table object with additional attributes derived from the model specification) is included in the returned object (default = TRUE). The model frame is required by the asreml summary, plot, resid and fitted methods.

In large analyses, the model frame is likely to be a large object. If model.frame is a character string, the model frame is saved in a file as an RDS object by a call to saveRDS(), and named by the supplied string with the extension .RDS. If the model frame is not included in the returned asreml object, this RDS file is searched for by the methods noted previously.

data — A data frame in which to interpret the variables named in fixed, random, sparse and residual. If the data argument is missing, the default is sys.parent(). The data frame is converted internally to a data.table object and returned as such by model.frame.

... — Additional arguments to asreml() directed at asreml.options(), such as: maxit, workspace, pworkspace, fixgammas, trace, aom.

Details

Models for asreml are specified symbolically in the formula objects: fixed, random, sparse and residual. A typical model has the form: response ~ terms, fixed only, or ~ terms for random, sparse and residual, where response is the (usually numeric) response vector and terms is a linear predictor for response.

The formulae objects are parsed in the context of the data frame, all internal data structures are constructed in **R** or compiled code, and the model is fitted by calls to the underlying Fortran REML routines (*Gilmour et al., 1995*). Variance models for random model terms are specified using *special* functions in the random and residual formulae. If not specified, the variance models default to (scaled) identity variance structures. A table of special model functions is included below; see the reference guide or appropriate vignette for further details and examples of their use. Some of these model functions require the formula arguments to be partially evaluated before the final model frame is computed; it is recommended that all names used in the formulae be resolvable in a data frame named by the data argument.

The terms in the fixed formula are re-ordered by default so that main effects precede interactions in increasing order. The option keep.order (see asreml.options) can be used to modify this behaviour.

A formula has an implied intercept term. To remove the intercept use: y ~ -1 + ... This is only effective in the fixed formula; in all other formula arguments any reference to the intercept is ignored. Note that currently there must be at least one fixed effect in the model.

In addition to the formal arguments, various options can be set with asreml.options; these are stored in an environment for the duration of the R session.

The library asreml uses either a "gamma" (ratio) or "sigma" (component) scale parameterization for estimation depending on the residual model specification. The current default for single-section analyses is the gamma parameterization if the error model specifies a correlation structure. In this case, all scale parameters are estimated as a ratio with respect to the residual variance, with correlation parameters unchanged. If the residual model specifies a variance structure then variance parameters are estimated on the sigma scale. For models with more than one residual section, asreml *always* estimates variance parameters on the sigma scale.

Value

An object of class asreml containing the results of the fitted linear model. Instances of generic methods such as plot(), predict() and summary() return various derived results of the fit. The methods resid(), coef() and fitted() extract some of its components. See asreml.object for details of the components of the returned list.

Special asreml model functions

Special model functions are used in asreml() formulae objects to create new or modify existing model terms, or more often to specify the variance model associated with one or more terms. These functions can be broadly categorized as: constructor-type functions, or identity, time-series, general-structure, metric-1D, metric-2D, and known relationship, general variance structures that span more than one term (str), and finally user-defined variance structures.

The special model functions that are available in asreml() model formulae are briefly described below; see the reference manual guide, relevant vignette or the main pages for selected functions and for more details or illustrations.

The arguments (symbols) used in the following descriptions are defined as:

obj	A factor in data.
n	Number of elements: length(levels(obj)).
p	Number of parameters estimated by the base function.
v	Number of parameters estimated by the homogeneous function form.
h	Number of parameters estimated by the heterogeneous function form.

Constructor model syntax type functions

Call	Description
con(obj)	Apply sum-to-zero constraints to factor obj.
C(obj, contr)	Define contrasts among the levels of obj from the coefficients in vector contr.
lin(obj)	Fit factor obj as a variate.
pow(obj, p, offset)	Create the term (offset + obj)^p.
pol(x, t)	Orthogonal polynomials to degree t; $-t$ omits the intercept polynomial.
leg(x, t)	Legendre polynomials to degree t; $-t$ omits the intercept polynomial.
spl(x, k)	The *random* component of a cubic spline; optionally k knot points.
dev(x)	Fit variate x as a factor; typically used for spline deviations.
ma(obj)	Form a moving average (order 1) design matrix from factor obj.
at(obj, l)	Form conditioning covariables for the levels in factor obj given in l.
dsum(~term | obj)	Form direct sum operation of term for the levels of obj. Used in residual to define multiple *section*
and(obj, t)	Add t times the design matrix for obj to the previous columns.
grp(obj)	Include the term defined by obj in the group argument in the model.
mbf(obj)	Include the covariates defined by obj in the obj argument as a factor.
sbs(obj)	Include the term defined in the prune argument in the model.
gpf(obj)	Include the term defined in the combine argument in the model.
uni(obj)	Include the term defined in the uid argument in the model.

Identity variance structure models (default)

Call	Description	p	v	h
id(obj)	identity	0	1	n

Time series type variance structure models

Call	Description	p	v	h
ar1(obj)	Autoregressive order 1	1	2	1+n
ar2(obj)	Autoregressive order 2	2	3	2+n
ar3(obj)	Autoregressive order 3	3	4	3+n
sar(obj)	Symmetric autoregressive order 1	1	2	1+n
sar2(obj)	Symmetric autoregressive order 2	2	3	2+n
ma1(obj)	Moving average order 1	1	2	1+n
ma2(obj)	Moving average order 2	2	3	2+n
arma(obj)	Autoregressive-moving average	2	3	2+n

General variance structure models

Call	Description	p	v	h
`cor(obj)`	Simple correlation	1	2	1+n
`corb(obj, b)`	Banded correlation; b bands	b	b+1	b+n
`corg(obj)`	General correlation	n(n-1)/2	1+n(n-1)/2	n(n+1)/2
`diag(obj)`	Diagonal variance	n		
`us(obj)`	Unstructured variance	n(n+1)/2		
`chol(obj, k)`	Cholesky order k	(k+1)(n-k/2)		
`cholc(obj, k)`	Cholesky C	(k+1)(n-k/2)		
`ante(obj, k)`	Antedependence order k	(k+1)(n-k/2)		
`sfa(obj, k)`	Factor analytic; k factors	kn+n		
`facv(obj, k)`	Factor analytic, covariance scale	kn+n		
`fa(obj, k)`	Sparse factor analytic	kn+n		
`rr(obj, k)`	Reduced rank variant of fa	kn+n		

Metric-based variance structure models in 1D or 2D

Call	Description	p	v	h
`exp(x, dist)`	Exponential (power) 1D	1	2	1+n
`iexp(x, y)`	Isotropic exponential (power) 2D	1	2	1+n
`aexp(x, y)`	Anisotropic exponential (power) 2D	2	3	2+n
`gau(x, dist)`	Gaussian power 1D	1	2	1+n
`igau(x, y)`	Isotropic Gaussian 2D	1	2	1+n
`agau(x, y)`	Anisotropic Gaussian 2D	2	3	2+n
`ieuc(x, y)`	Isotropic Euclidean 2D	1	2	1+n
`lvr(x, dist)`	Linear variance 1D	1	2	1+n
`ilv(x, y)`	Isotropic linear variance 2D	1	2	1+n
`sph(x, y)`	Isotropic spherical 2D	1	2	1+n
`cir(x, y)`	Isotropic circular 2D	1	2	1+n
`mtrn(x, y, ...)`	Matern variance 2D	*	*	*

*See the reference manual for an extended discussion of the Matern variance class.

Known relationship variance structure models

Call	Description
`vm(obj, source, singG)`	Create a term based on `obj` with known variance structure in `source`.
`ide(obj)`	Identity term based on `obj` with levels as for `vm(obj)`

General variance structures models

Call	Description
`str(~terms, ~model)`	Apply the direct product variance structure in `~model` to the set of terms in `~terms`.

User-defined variance structure models

Call	Description
`own(obj, fun, init, type)`	Call `fun` with the parameter estimates in `init` to compute the variance matrix and its derivatives.

References

A R Gilmour, R Thompson, B R Cullis (1995). "AI, An Efficient Algorithm for REML Estimation in Linear Mixed Models." *Biometrics*, **51**, 1440-1450. M G Kenward, J H Roger (1997). "The Precision of Fixed Effects Estimates from Restricted Maximum Likelihood." *Biometrics*, **53**, 983-997.

See Also

`asreml.options` `asreml.object` `family_dist`

Examples

```
## Not run:
data(oats)
oats.asr <- asreml(yield ~ Variety*Nitrogen, random = ~ Blocks/Wplots, data = oats)

## End(Not run)
```

---

asreml.license.activate

Activates an `asreml` *license*

Description

Prompt the user for an activation code and activate the `asreml` license.

Usage

```
asreml.license.activate(acode = NULL, ...)
```

Arguments

`acode`	An activation code to activate an `asreml` license. If not supplied then a prompt will be displayed asking for the code.
`...`	Additional arguments, for example core_version.

Details

It uses a supplied activation code to activate an `asreml` license. IT Security measures can sometimes block connections when trying to activate a license.

Value

The result of the license activation is silently returned.

`asreml.license.deactivate`

Deactivates the current `asreml` *license*

Description

This function will make an http request to the license activation server causing it to revoke the `asreml` license fulfillment associated with the current system.

Usage

```
asreml.license.deactivate(...)
```

Arguments

`...`	Additional arguments, for example core_version.

Details

IT Security measures can sometimes block connections when trying to deactivate an `asreml` license.

Value

The result of the license deactivation is silently returned.

`asreml.license.offline`

Request an `asreml` *license to be used offline*

Description

This function will request the http to use the `asreml` license for a number of days. The default is 2 days and may be set to a maximum of 30 days.

Usage

```
asreml.license.offline(how_many_days = 2, ...)
```

Arguments

`how_many_days`	An integer specifying the number of days to use the `asreml` license offline (default = 2).
`...`	Additional arguments, for example core_version.

Details

You can call this method to request to use one of the license entitlements fully offline. When a license is set to be used offline it will reduce the number of online license entitlements by one. A license can be used offline for a maximum of 30 days and can be returned at any point during that period. You can return a license to online usage by using the function `asreml.license.online()`. This feature is not available when running `asreml` on a virtual machine.

`asreml.license.online` *Request an* `asreml` *license to be used online*

Description

This functions will return a license that is currently offline to be used online.

Usage

```
asreml.license.online(...)
```

Arguments

`...`	Additional arguments, for example core_version.

Details

If `asreml` is being used offline you can return the license to online usage by calling this method. This feature is not available when running `asreml` on a virtual machine.

`asreml.license.status` *Check the status of the* `asreml` *license*

Description

Function that checks the status of the `asreml` license and reports any errors.

Usage

```
asreml.license.status(quiet = FALSE, task = "checkout", json = "", ...)
```

Arguments

`quiet`	If `FALSE`, a string containing the license details is echoed to the console (default = `FALSE`).
`task`	A character string specifying the task used when checking the license.
`json`	A character string specifying additional parameters for the license check in JSON format.
`...`	Additional arguments, for example core_version.

Details

It checks the status of the asreml license and displays details about the license owner, expiry date and time remaining.

Value

The result of the license status is silently returned.

---

asreml.object — *This (S3) class object contains a fitted linear mixed model from the* asreml() *function*

Description

Objects of this class have methods for the generic functions wald(), coef(), fitted(), plot(), predict(), resid(), summary() and update().

Value

A list object with class asreml; the following components are included in a valid asreml object:

- loglik: The log-likelihood at completion of the asreml call.
- vparameters: The vector of variance components estimates from the model fit.
- vparameters.con: A numeric vector identifying the boundary constraint applied to each variance parameter at termination. Common values are 1, 3 and 4 for **P**ositive, **U**nconstrained and **F**ixed, respectively.
- vparameters.type: A numeric vector identifying the variance parameter types.
- vparameters.pc: Percentage change in gammas parameters over the last iteration.
- score: The score vector of length number of random parameters.
- coefficients: A list with three components named fixed, random and sparse containing the solutions to the mixed model equations corresponding to the E-BLUEs of the fixed effects, the E-BLUPs of the random effects, and the solutions corresponding to the sparse fixed effects, respectively. The coefficients are labelled by a concatenation of factor name and level separated by "_".
- vcoeff: A list with three components named fixed, random and sparse containing the unscaled variances of the coefficients. The actual variances are calculated as vcoeff*object\$sigma2 and returned by the summary function.
- predictions: If predict is not NULL, a list object with components pvals, sed, vcov and avsed. The predictions component only is returned by the predict method for asreml objects.
- fitted.values: A vector containing the fitted values from the model, obtained by transforming the linear predictors by the inverse of the link function.
- linear.predictors: The linear fit on the link scale.

- residuals: A single-column matrix containing the residuals from the model.
- hat: The diagonal elements of the matrix $WC^{-1}W^T$, the *extended* hat matrix. This is the linear mixed effects model analogue of $X(X^TX)^{-1}X^T$ for ordinary linear models.
- sigma2: The REML estimate of the scale parameter.
- deviance: The deviance from the fit.
- nedf: The residual degrees of freedom, length(y) - rank(X).
- nwv: The number of working variables.
- noeff: A vector containing the number of effects for each term.
- yssqu: A vector of incremental sums of squares for (dense) fixed terms.
- ai: The inverse average information matrix of the variance parameters. A Matrix class object, sub-class dspMatrix is returned.
- Cfixed: Reflexive generalized inverse of the coefficient matrix of the mixed model equations relating to the dense fixed effects (if asreml.options()$Cfixed = TRUE, a matrix of class Matrix, sub-class dspMatrix is returned).
- Csparse: If asreml.options()$Csparse is not NULL, the non-zero elements of the reflexive generalized inverse matrix of the coefficient matrix for the sparse stored model terms nominated in the Csparse formula. A matrix in triplet form giving the row, column and non-zero elements.
- design: The design matrix as a sparse Matrix of class dgCMatrix if asreml.options(design = TRUE).
- call: An image of the asreml function call.
- trace: A numeric matrix recording the convergence sequence for each random component, as well as the log-likelihood, residual variance and residual degrees of freedom.
- license: A character string containing the license information. The string has embedded new-line characters and is best formatted through cat().
- G.param: A list object containing the constraints and final estimates of the variance parameters relating to the random part of the model. This object may be used as the value of the G.param argument to provide initial parameter estimates to asreml.
- R.param: A list object containing the constraints and final estimates of the variance parameters relating to the error structure of the model. This object may be used as the value of the R.param argument to provide initial parameter estimates to asreml.
- formulae: A list object containing the fixed, random, sparse and residual formula arguments to asreml.
- factor.names: A character vector of term names appearing in the model.
- meff: Regressor scores (marker effects) if nominated in the mef list argument.
- mf: The model frame with the data as a data.table object with numerous attributes from the model specification. Inspect names(attr(object$mf)) for details.

asreml.options *Set less frequently used* asreml() *options*

Description

In asreml.options() the less frequently used settings are set per session outside asreml() in an options environment. With no arguments, asreml.options() returns a list of settings that can be altered.

Usage

```
asreml.options(...)
```

Arguments

about	If TRUE, the date packaged is printed for identification (default = FALSE).
ai.loadings	Controls modification to Average Information (AI) updates of loadings in extended factor-analytic (fa(, k)) models. After asreml calculates updates for variance parameters, it checks whether the updates are reasonable and sometimes reduces them over and above any step.size shrinkage. The extra shrinkage has two levels: loadings that change sign are restricted to doubling in magnitude; and if the average change in magnitude of loadings is greater than 10-fold, they are all shrunk. Unless the user specifies constraints, asreml sets them and rotates the loadings each iteration. When ai.loadings = i is specified ($i = -1$ is no action), it also prevents AI updates of some loadings during the first i iterations. For $f > 1$ factors, only the last factor is estimated (conditional on the earlier ones) in the first $f - 1$ iterations. Then pairs, including the last, are estimated until iteration i (default = 0).
ai.penalty	This refers to the algorithm for updating loadings in factor analytic (fa) models. The present strategy modifies the average information matrix by increasing the diagonal elements pertaining to loadings by a percentage, p. The default is to start with $p = 10\%$ and reduce it by 1 or 2% each iteration down to 1%. If the starting values are poor, 10% may not be a sufficient initial retardation. If it appears the updates are unreasonable, the value of p is increased by 10%. After the penalty has reduced to 1%, it is further reduced to 0.2%. ai.penalty can be set to 0 if desired (default = 10, corresponding to 10%).
ai.sing	If TRUE, forces continuation if a singularity is detected in the average information matrix. Variance components are reported to help identify singularity but these are often incorrect (default = FALSE).
aodev	If TRUE, return an analysis of deviance (default = FALSE).
aom	If TRUE, return standardized conditional residuals and standardized conditional BLUPs in the aom component of the asreml object (default = FALSE).
Cfixed	If TRUE, return the computed part of the C^{-1} matrix in component Cfixed. The inverse coefficient matrix is fully formed for terms in the dense set (default = FALSE).

`colourise` If `TRUE`, the header text produced by methods such as `wald` and `predict` will be displayed in a different colour if supported by the output terminal device (default = `TRUE`).

`Csparse` If a formula is specified, return the computed part of C^{-1} for those terms given in the formula. The library `asreml` does not compute the whole of C^{-1}, only that which is sufficient to calculate the REML solution (default = `Csparse = ~NULL`).

`debug` Return internal data structures for helping debugging (default = `FALSE`).

`dense` Include the equation(s) for the term(s) in the formula in the dense set. This results in faster processing if the term is associated with a known dense inverse relationship matrix (default = `dense = ~NULL`).

`design` If `TRUE`, return the design matrix for component `design` of the `asreml` object. This option might be used, for example, in generating design matrices that can then be used in post-processing (default = `FALSE`).

`drop.unused.levels` If `TRUE`, levels of simple factors that do not appear in the data are dropped (default = `TRUE`).

`eqorder` Set the algorithm used for ordering the mixed-model equations prior to solution. The argument `eqorder = -1` processes the equations in *user* order; generally this will run much slower, if at all, in real time for large analyses (default = `3`).

`extra` Forces other `mod(extra, maxit)` extra iterations after apparent convergence (default = `0`).

`fail` If `"hard"`, fatal errors will terminate execution, otherwise if `"soft"` such conditions will be reported as warnings, allowing testing runs, for example, to continue. In both cases the `converge` component of the `asreml` object will be set to `FALSE` and the results will be erroneous (default = `"hard"`).

`fixgammas` If `TRUE`, all variance parameters are constrained to be fixed at their starting values (default = `FALSE`).

`font.scale` Scale axis text and labels (relative to the `asreml` default settings) in the graphs generated by `plot()` (default = `1.0`).

`gammaPar` If `TRUE`, single-section-models will be fitted using the `gamma` parameterization irrespective of whether the `residual` formula specifies a correlation or variance model. The default behaviour for single-section models is to fit on the `gamma` scale if the `residual` formula specifies a correlation structure, and on the `sigma` scale if the `residual` formula specifies a variance structure (default = `FALSE`).

`glmminloop` Set the number of inner iterations performed in an iteratively weighted least squares analysis. These estimate the effects in the linear model for the current set of variance parameters; outer iterations are the average information updates to the variance parameters. The default is to perform 4 inner iterations in the first round and 2 in subsequent rounds of the outer iteration. Set to 2 or more to increase the number of inner iterations (default = 1).

`grid` A logical vector of length 1 or `length(design.points)` (see predict) controlling the expansion of coordinates for two-dimensional kriging. For a given term, the coordinates for prediction in two dimensions (x, y) are given as a list of two vectors or a two-column matrix component of `design.points`. If `TRUE`, the

coordinates are expanded to form an (x, y) grid of all possible combinations, otherwise the columns of the matrix are taken in parallel (default = TRUE).

`keep.order` If TRUE, the order of terms in the `fixed` formula is retained. Set to TRUE if the special model function `and()` is present (default = FALSE).

`knots` The number of knot points for spline terms to consider. For a variate `x`, the number of knot points is `min(length(unique(x)), knots)` (default = 50).

`license_id` If 0, a new license_id is created for identification (default = 0).

`maxit` Maximum number of iterations to stop fitting (default = 13).

`nsppoints` Value that influences the number of points used when predicting splines and polynomials. The design matrix generated by the `pol(x)` and `spl(x)` functions are modified to include extra rows for points used in prediction. The range of `x` is divided by `nsppoints - 1` to give a step size i. For each point p in x, a predict point is inserted at $p + i$ if there is no data value in the interval $[p, p + 1.1i]$.

The argument `nsppoints` is ignored if the `predict.asreml()` argument `design.points` is set (or the `design.points` component of the `predict` list argument to `asreml()` is not empty). This process also affects the number of levels identified by `dev(x)` (default = 21).

`oscillate` If TRUE, the test for oscillating log-likelihood is implemented (default = TRUE).

`pworkspace` Sets the workspace needed by the `predict()` method; follows the same convention as `workspace`. Ignored if the `predict` argument to `asreml()` is not set. Note that the total workspace used for prediction is: `workspace` + `pworkspace` (default = "128mb").

`pxem` Selection of the (PX)EM update strategy for unstructured (`us`) variance models when average information updates cause them to be non-positive definite (see `uspd`) (default = 1). Valid options are:

Option	Action
1	standard EM + 10 local EM steps
2	standard EM + 10 local PXEM steps
3	standard EM + 10 local EM steps*
4	standard EM + 10 local PXEM steps*
5	standard EM only
6	single local PXEM
7	standard EM + 1 local EM step
8	standard EM + 1 local PXEM step

*Options 3 and 4 cause all `us` structures to be updated by (PX)EM if any particular one requires EM updates.

`random.order` If option "noeff" the terms in the `random` and `sparse` formulae are reordered in increasing number of effects. This is almost always desirable, especially if the stratum variance decomposition is required. Other options are "user" to retain the order given in the model specification, or "R" for the default **R** rules (default = "noeff").

`rotate.fa` If FALSE, `asreml()` initially constrains the first $k - 1$ loadings for higher order ($k > 1$) factors in factor analytic models to zero. If constraints are not set

for factor analytic models with more than one factor, `asreml()` will set them internally and rotate the loadings each iteration (`rotate.fa = TRUE`). This option also modifies the action of `update.Gcon` such that rotation, if specified, is applied on an update (default = `FALSE`).

`scale` Overall scale parameter (default = `1.0`).

`score` If `TRUE`, the score vector is returned in a component `score` of the `asreml` object (default = `FALSE`).

`spline.scale` When forming a design matrix for a `spl()` term, a standardized scale is used. Setting `spline.scale = 1` forces `asreml` to use the scale of the variable. The value `-1` is recommended in most cases (default = `-1`).

`spline.step` A list with components named `spl`, `dev` and `pol` specifying the resolution for spline deviations and polynomial functions, respectively. Points closer together than `1/spline.step` of the range will be treated as a single point (default `spline.step = list(spl = 10000, dev = 10000, pol = 10000)`).

`step.size` Value for updating shrinkage factor. It reduces the update step sizes of the variance parameters. The step size is incremented each iteration to a maximum of 1.0 (default = `0.316`).

`threads` The maximum number of threads to be used with OMP parallel processing. `asreml` will use all threads available up to the maximum number specified. The value `-1` will use all available threads (default = `-1`).

`tol` A vector of length two that modifies the sensitivity of `asreml` to detect singularities in the mixed model equations. This is intended for the rare occasions when singularities are detected after the first iteration (default = `c(0, 0)`).

Normally a singularity is declared if the adjusted sum of squares of a covariable is less than e, or less than the *uncorrected sum of squares* $\times e$, where $e = 10^{-8}$ in the first iteration and 10^{-10} thereafter. If `tol = c(a, b)`, e is scaled by 10^a for the the first iteration, and 10^b for subsequent iterations. Once a singularity is detected, the corresponding equation is dropped (forced to be zero) in subsequent iterations.

If the problem of later singularities arises because of the low coefficient of variation of a covariable, it may be advisable to centre and rescale the covariable. If the degrees of freedom are correct in the first iteration, the problem lies with the variance parameters and a different variance model (or constraint) is needed.

`trace` If `TRUE`, convergence monitoring of the current fit is reported in the console (default = `TRUE`).

`update.Gcon` If `TRUE`, the constraint status of variance parameters in the `G.param` list component on termination is updated; this may influence subsequent updates to the model fit (default = `TRUE`).

`update.Rcon` If `TRUE`, the constraint status of variance parameters in the `R.param` list component on termination is updated; this may influence subsequent updates to the model fit (default = `TRUE`).

`update.step.size`

Value for updating shrinkage factor to use in a call to `update()`. If ignored it is set to `0.316` or if `step.size` is explicitly specified on the `update()` call; otherwise the shrinkage factor is set to `update.step.size` in the call to `asreml()` constructed by `update()` (default = `0.316`).

use.blas — If "standard", then asreml will make use of linear algebra routines compiled into the asreml package, otherwise the version of BLAS / LAPACK from R will be used (default = "standard").

uspd — If TRUE, set the boundary constraint for each parameter in unstructured variance models to "P". Under these conditions, asreml checks whether the updated matrix is positive definite; if not, the average information update is replaced with an EM update (see pxem) (default = TRUE).

workspace — Sets the workspace for the core REML routines in the form of a number optionally followed directly by a valid measurement unit. Valid units are kb, mb or gb; if no units are given then the value is interpreted as double-precision words (groups of 8 bytes) (default = "128mb").

deepcopy — Only relevant if the object passed to the data argument in asreml is of class data.table. If TRUE, a copy of data is created in memory within the function scope; in this case, the original object passed to data is not modified. If FALSE, the original object is modified by reference. Setting this option to FALSE is useful when the original data object is large, leading to less memory usage (default = FALSE).

Details

Arguments are in the form name = value, where name is the name of the option to set.

---

asreml.read.table *Read in a data frame*

Description

Function that reads in a file in table format and creates a data frame with the same number of rows as there are lines in the file, and the same number of variables as there are fields in the file. Variables whose names begin with a capital letter are converted to factors.

Usage

```
asreml.read.table(...)
```

Arguments

... — Arguments to be passed to read.table.

`asr_varioGram` *Empirical variogram*

Description

Calculates the empirical variogram from regular or irregular one- or two-dimensional data.

Usage

```
asr_varioGram(
  x,
  y,
  z,
  composite = TRUE,
  model = c("empirical"),
  metric = c("euclidean", "manhattan"),
  angle = 0,
  angle.tol = 180,
  nlag = 20,
  maxdist = 0.5,
  xlag = NA,
  lag.tol = 0.5,
  grid = TRUE,
  ...
)
```

Arguments

`x`	Numeric vector of *x* coordinates, may also be a matrix or data frame with two or three columns. If `ncol(x)` is three, the columns are taken to be the *x* and *y* coordinates and the response (*z*), respectively. If `ncol(x)` is two, the columns are taken to be the *x* coordinates and the response, respectively. In this case the *y* coordinates are generated as `rep(1, nrow(x))`.
`y`	Numeric vector of *y* coordinates.
`z`	The response vector.
`composite`	To be used for data on a regular grid. If `TRUE`, the average of the variograms in quadrants *(x, y)* and *(x, -y)* is returned. Otherwise, both variograms are returned and identified as quadrants 1 and 4 (default = `TRUE`).
`model`	At the present, it can only be `"empirical"`.
`metric`	The distance between *(x, y)* points. Options are: `"euclidean"` or `"manhattan"` (default = `"euclidean"`).
`angle`	A vector of directions. Angles are measured in degrees anticlockwise from the *x* axis (default = `0`).
`angle.tol`	The angle subtended by each direction. That is, an arc `angle +/- angle.tol/2` (default = `180`, which gives an omnidirectional variogram).

nlag	The maximum number of lags (default = 20).
maxdist	The fraction of the maximum distance to include in the calculation. The default is half the maximum distance in the data: 0.5.
xlag	The width of the lags. If missing, xlag is set to maxdist/nlag.
lag.tol	The distance tolerance (default = 0.5).
grid	If FALSE, forces polar variograms if *(x, y)* specifies a regular grid (default = TRUE).
...	Additional arguments, for example core_version.

Details

For one-dimensional data the *y* coordinates need not be supplied and a vector of ones is generated. The function identifies data on a complete regular array and in such cases only computes polar variograms if grid = FALSE. The data are assumed sorted with the *x* coordinates changing the fastest; the data are sorted internally if this is not the case.

Value

A data frame including the following components:

- x: The original *x* coordinates.
- y: The original *y* coordinates.
- gamma: The variogram estimate.
- distance: The average distance for pairs in the lag.
- np: The number of pairs in the lag.
- angle: Direction if not a regular grid.

References

W Webster, M A Oliver (2001). *Geostatistics for Environmental Scientists*. John Wiley: West Sussex.

---

asuv *Univariate data frame*

Description

Function that makes a univariate data frame from a multi-variate one.

Usage

```
asuv(
  stack,
  data,
  response = "y",
  traitName = "trait",
  traitsWithinUnits = TRUE
)
```

Arguments

stack	A character or numeric vector identifying the columns that form the multi-variate response. These columns are concatenated into a vector.
data	The multi-variate data frame.
response	A character string to be used as the name of the concatenated response vector (default = "y").
traitName	A character string to name the resulting column of multi-variate response names (default = "trait").
traitsWithinUnits	The sort order of the returned data frame. If TRUE, orders the data as multi-variate response traits nested within experimental units (default = TRUE).

Details

This is a stack operation on the multi-variate data frame based on the nominated response columns. These are concatenated into a vector and the remaining columns of the input data frame repeated accordingly. A "trait" column is created from the names of the multi-variate response columns.

Value

The stacked data frame.

---

asv *Approximate stratum variances*

Description

Generic function to calculate approximate stratum variances, degrees of freedom and coefficients for variance component models. The available method is for asreml class objects.

Usage

```
asv(object, ...)
```

Arguments

object	An object of class asreml.
...	Arguments to asv.asreml

Details

For linear mixed effects model, it is often possible to consider a natural ordering of the variance component parameters, including the residual. Based on an idea due to *Thompson, 1980*; here, asreml computes approximate stratum degrees of freedom and stratum variances by a modified Cholesky diagonalisation of the average information matrix.

See Also

asv.asreml

---

asv.asreml *Approximate stratum variances*

Description

Generic function to compute approximate stratum variances, degrees of freedom and coefficients for variance component models.

Usage

```
## S3 method for class 'asreml'
asv(object, order = c("as.is", "noeff"), vvp = FALSE, ...)
```

Arguments

object	An object of class asreml.
order	The sequence in which to consider the random terms. Options are "as.is" for the order as given in the asreml object, or "noeff" to be ordered by increasing number of effects, respectively. Note that the default ordering for random terms in asreml is "noeff" (default = "as.is").
vvp	If TRUE, the inverse of the average information matrix for the variance components (sigma scale) is returned (default = FALSE).
...	Any additional arguments.

Details

For linear mixed effects model, it is often possible to consider a natural ordering of the variance component parameters, including the residual. Based on an idea due to *Thompson, 1980*; here, asreml computes approximate stratum degrees of freedom and stratum variances by a modified Cholesky diagonalisation of the average information matrix.

Value

If vvp = FALSE, a rectangular matrix containing the degrees of freedom, approximate variances and the component coefficients for each random term is provided.

If vvp = TRUE, a list of length two containing the matrix of approximate stratum variances and the inverse average information matrix of the variance components in list components asv and vvp, respectively, is provided.

References

R Thompson (1980). "Maximum likelihood estimation of variance components." *Series Statistics*, **11**(4), 545-561.

binnor *Footrot scores on lambs*

Description

Incidence of two foot shape classes on 2513 lambs. The feet of 2513 lambs born in 1980 and 1981 from 5 mating groups were scored in two foot shape classes: 1) all four feet are normal, and 2) one foot is deformed. Two indicator variables were also recorded for the presence of the disease conditions scald and rot.

Usage

```
binnor
```

Format

A data frame with 10 columns and 2513 rows:

- record: Record number, integer values
- year: Cross year, an integer with values 1 and 2
- Grp: Group, factor with 5 levels
- Sex: Sex, factor with 2 levels
- Sire: Sire number, factor with 18 levels
- score5: Incidence of foot shape class 2
- score4: Incidence of foot shape class 1
- scald: Incidence of scald disease
- rot: Incidence of rot disease, binary response
- norm: A random Normal variate response

References

A R Gilmour, R D Anderson, A L Rae (1985). "The Analysis of Binomial Data by a Generalized Linear Mixed Model." *Biometrika*, **72**, 593-599.

captions *Axis captions*

Description

Vector of axis labels for each residual type.

Usage

```
captions
```

Format

A named vector of axis label strings.

cheese *Cheese tasting panel (multivariate form)*

Description

Tasting category counts on 4 cheeses. Four cheeses were scored on a 9-point scale by 52 judges.

Usage

```
cheese
```

Format

A data frame with 11 columns and 4 rows:

- Cheese: Cheese type, factor with 4 levels
- cat1: Taste category 1 counts
- cat2: Taste category 2 counts
- cat3: Taste category 3 counts
- cat4: Taste category 4 counts
- cat5: Taste category 5 counts
- cat6: Taste category 6 counts
- cat7: Taste category 7 counts
- cat8: Taste category 8 counts
- cat9: Taste category 9 counts
- tot: Total counts for each cheese

References

P McCullagh, J A Nelder (1989). *Generalized Linear Models*. CRC Press.

cheese.cat *Cheese tasting panel (incidence form)*

Description

Tasting incidences on four cheeses. Four cheeses were scored on a 9-point scale by 52 judges. This is the cheese data in *incidence* form, where each taste category and cheese combination is recorded.

Usage

```
cheese.cat
```

Format

A data frame with 2 columns and 208 rows:

- Taste: Taste category (1-9), factor with 9 levels
- Cheese: Cheese type, factor with 4 levels

References

P McCullagh, J A Nelder (1989). *Generalized Linear Models*. CRC Press.

chkPed *Check a pedigree*

Description

Checks a pedigree for consistency, adds missing founders and sorts so that founders appear before individuals.

Usage

```
chkPed(
  pedigree,
  fgen = list(character(0), 0.01),
  gender = character(0),
  mv = c("0", "NA"),
  core_version = NULL
)
```

Arguments

pedigree	A data frame where the first three columns correspond to the identifiers for the individual, male parent and female parent, respectively. Founders or unknown parents use 0 (zero) or NA in the parental columns.
fgen	An optional list of length two where `fgen[[1]]` is a character string naming the column in `pedigree` that contains the level of selfing or the level of inbreeding of an individual. In `pedigree[,fgen[[1]]]`, 0 indicates a simple cross, 1 indicates selfed once, 2 indicates selfed twice, etc. A value between 0 and 1 for a base individual is taken as its inbreeding value. If the pedigree has implicit individuals (they appear as parents but not as individuals), they will be assumed base non-inbred individuals unless their inbreeding level is set with `fgen[[2]]`, where `0 < fgen[[2]] < 1` is the inbreeding level of such individuals.
gender	An optional character string naming the column of `pedigree` that codes for the gender of an individual. `pedigree[,gender]` is coerced to a factor and must only have two (arbitrary) levels, the first of which is taken to mean "male". An inverse relationship matrix is formed for the X chromosome as described by *Fernando and Grossman, 1990* for species where the male is XY and the female is XX.
mv	A character vector of missing value indicators; elements of `pedigree` that exactly match any of the members of `mv` are treated as missing. The values passed should be used in addition to `"0"` and `"NA"` (default = `"0", "NA"`).
core_version	The version of the algorithmic core to use, not usually required.

Value

The pedigree in a data frame, reordered and expanded if necessary.

---

coef.asreml *Extract model coefficients*

Description

Extracts model coefficients (solutions for BLUEs or BLUPs) from an `asreml` object.

Usage

```
## S3 method for class 'asreml'
coef(object, list = FALSE, pattern = ~NULL, ...)
```

Arguments

object	An `asreml` object.
list	If `TRUE`, the coefficients are returned in a named list whose length is the number of terms in the model (default = `FALSE`).
pattern	A term in the model as a one-sided formula. A regular expression is constructed from `pattern` to extract a subset of coefficients.
...	Additional arguments.

Value

If neither `pattern` nor `list` are set, then a list of length two with the following components:

- `fixed`: Solutions to the mixed model equations for the fixed terms.
- random: E-BLUPs for the effects in the random model.

where each component is a matrix with a `dimnames` attribute.

If `list = TRUE`, a list object where each component is a single-column matrix corresponding to a term in the model is returned. Otherwise, a single-column matrix of effects as specified by `pattern`.

Examples

```
## Not run:
data(oats)
oats.vsr <- asreml(yield ~ Variety*Nitrogen, random = ~ Blocks/Wplots, data = oats)
coef(oats.vsr)
coef(oats.vsr, list = TRUE)
coef(oats.vsr, pattern = ~Blocks:Wplots)

## End(Not run)
```

---

complex_vs *Variance structure spanning several model terms*

Description

General variance structure specifying a set of terms to be included in the random argument that collectively will have an associated variance model.

Usage

```
str(form, vmodel)
```

Arguments

`form`	A model formula included verbatim in the `asreml()` `random` argument.
`vmodel`	A direct product variance model for the set of terms given in `form`.

Details

Typically, a variance structure applies to an individual term (main effect or interaction) in the linear model, and there is no covariance between random model terms. Sometimes it is appropriate to include a covariance, such as random coefficients regression. In such cases it is essential that the model terms be contiguous and that the variance structure defined is the structure required across all terms in the set.

The model terms in `form` are consequently not reordered. While `asreml` will check the overall size of the included terms, it cannot check that the order of effects matches the structure definition in `vmodel`; care must be taken to ensure this is correct. Check that the terms are conformable by considering the order of the fitted effects and ensuring the first term of the direct product in `vmodel` corresponds to the outer factor in the nesting of the effects in `form`.

---

coop *Coopworth lamb data*

Description

Measurements on 7043 lambs, who were the progeny of 92 sires and 3561 dams, produced from 4871 litters over 49 flock-year combinations.

Usage

```
coop
```

Format

A data frame with 13 columns and 7043 rows:

- tag: Lamb ID, factor with 7043 levels
- sire: Sire ID, factor with 92 levels
- dam: Dam ID, factor with 3561 levels
- grp: Flock year group, factor with 49 levels
- sex: Sex, factor with 2 levels
- brr: Birth rearing rank, factor with 4 levels
- litter: Litter ID, factor with 4871 levels
- age: Age, numeric covariate
- wwt: Weaning weight
- ywt: Yearling weight
- gfw: Greasy fleece weight
- fdm: Fibre diameter
- fat: Ultrasound fat depth at the C site

decode_json *Set the package global variable*

Description

It sets the (package) global variable `.asremlEnv$asr_lm[[core_id]]$r_license`

Usage

```
decode_json(rv, core_id)
```

Arguments

`rv`	A value used to set `.asremlEnv$asr_lm[[core_id]]$r_license`. The argument `rv` is either a JSON string, in which case it is parsed or an R object, in which case it isn't.
`core_id`	ID of the core for which r_license is being set.

ebay *Ebay bids*

Description

Hypothetical bids within 15 auctions for Ebay. These are nested correlated observations

Usage

```
ebay
```

Format

A data frame with 3 columns and 2500 rows:

- seq: Sequential record number, integer value
- auc: Auction ID, integer value
- val: Bid value, numeric value

`fa.init` *Factor analytic initial values*

Description

Obtains initial parameter values for factor analytic models from the observed variance-covariance matrix.

Usage

```
fa.init(data, model, scale = 1, core_version = NULL)
```

Arguments

`data`	A data frame containing the response and the base factors in the factor analytic (compound) model term.
`model`	A two-sided formula with the response on the left and a first order interaction with a factor analytic model term on the right. Valid factor analytic model functions are `"sfa"`, `"facv"`, `"fa"` or `"rr"` with an optional argument `k` specifying the order of the factor analytic model.
`scale`	Scales the variance-covariance matrix prior to calculating the parameter estimates (default = `1.0`).
`core_version`	The version of the algorithmic core to use, not usually required.

Value

A numeric vector of specific variances followed by the loadings if the factor analytic function is `"fa"` or `"rr"`, otherwise a numeric vector of loadings followed by the specific variances.

Examples

```
## Not run:
init <- fa.init(data, fa(site,2):variety)

## End(Not run)
```

family_dist *GLM (and GLMM) family objects for* asreml

Description

Family functions specify the details of the models accepted by the family argument to asreml used for fitting generalized linear models (GLMs) and/or generalized linear mixed models (GLMMs).

Usage

```
asr_gaussian(link = "identity", dispersion = NA)

asr_Gamma(link = "inverse", dispersion = 1, phi = 1)

asr_binomial(link = "logit", dispersion = 1, total = NULL)

asr_negative.binomial(link = "log", dispersion = 1, phi = 1)

asr_poisson(link = "log", dispersion = 1)
```

Arguments

link	A character string identifying the link function; options and valid names are: • Gaussian: "identity", "log", "inverse" (default = "identity") • Gamma: "identity", "log", "inverse" (default = "inverse") • binomial: "logit", "probit", "cloglog" (default = "logit") • negative.binomial: "identity", "log", "inverse" (default = "log") • poisson: "identity", "log", "sqrt" (default = "log")
dispersion	If NA, the dispersion parameter is estimated, otherwise it is fixed at the nominated value (default = NA for Gaussian and inverse Gaussian models, for the other families default = 1.0).
phi	The known value of the additional parameter phi.
total	A character string or name giving the column in data containing the total counts.

Value

A list of functions and expressions needed by the family argument.

Functions

- asr_gaussian(): The Gaussian model/distribution (default).
- asr_Gamma(): The gamma model/distribution.
- asr_binomial(): The binomial model/distribution. If the response is between 0 and 1 it is interpreted as the proportion of successes, otherwise, if not a binary (0,1) variate, it is interpreted as counts of successes; the total number of cases is given by the total argument. If total = NULL, a binary (0,1) response is expected.

- `asr_negative.binomial()`: The negative-binomial model/distribution.
- `asr_poisson()`: The poisson model/distribution.

`fitted.asreml` *Extract fitted values*

Description

Extracts fitted values from an `asreml` object.

Usage

```
## S3 method for class 'asreml'
fitted(
  object,
  type = c("response", "link"),
  value = c("cell", "cumulative"),
  ...
)
```

Arguments

`object`	An `asreml` object.
`type`	If `"link"`, the fitted values are on the link scale. Otherwise, if `"response"`, these are obtained by transforming the linear predictors by the inverse link function.
`value`	If `type ="response"` and `object` is a multinomial threshold model, then `"cell"` returns cell probabilities, else if `"cumulative"` then cumulative probabilities are returned.
`...`	Additional arguments.

Value

A numeric vector of fitted values.

Examples

```
## Not run:
data(oats)
oats.vsr <- asreml(yield ~ Variety*Nitrogen, random = ~ Blocks/Wplots, data = oats)
fitted(oats.vsr)

## End(Not run)
```

gamma2sigma *Transforms variance components and AI matrix from gamma to sigma scale*

Description

Some of the functions (summary.asreml and vpredict) require the components and the inverse of the average information (AI) matrix to be in sigma scale. If the model is in gamma, it has to be transformed This function can be used for that purpose.

Usage

```
gamma2sigma(
  object,
  where.sigma2 = attr(is.gamma(object), "where.sigma2"),
  what = c("varcomp", "ai"),
  ...
)
```

Arguments

object	An asreml object.
where.sigma2	A numeric value indicating the index of the sigma2 component in the vparameters object of the list object of class asreml (default = attr(is.gamma(object), "where.sigma2")).
what	A character vector indicating which object to obtain. Options are varcomp (the variance components) and ai (the inverse of the average information matrix) (default = (c("varcomp", "ai"))).
...	Additional arguments to asreml.options.

Value

A list with variance components (if requested; varcomp) and the inverse of the AI matrix (if requested; ai), both in sigma scale.

Examples

```
## Not run:
asr.oats <- asreml(
  fixed = yield ~ Variety * Nitrogen,
  random = ~ Blocks / Wplots,
  data = oats)

gamma2sigma(asr.oats)

## End(Not run)
```

general_vs *General structure variance models*

Description

General correlation and variance structures. The class of general variance models includes the simple, banded and general correlation models (cor, corb, corg), the diagonal, unstructured, Cholesky and antedependence variance models (diag, us, chol, cholc, ante) and the factor analytic structures (sfa, facv, fa).

Usage

```
cor(obj, init = NA)

corv(obj, init = NA)

corh(obj, init = NA)

corb(obj, b = 1, init = NA)

corbv(obj, init = NA)

corbh(obj, init = NA)

corg(obj, init = NA)

corgv(obj, init = NA)

corgh(obj, init = NA)

diag(obj, init = NA)

us(obj, init = NA)

chol(obj, k = 1, init = NA)

cholc(obj, k = 1, init = NA)

ante(obj, k = 1, init = NA)

sfa(obj, k = 1, init = NA)

facv(obj, k = 1, init = NA)

fa(obj, k = 1, init = NA)

rr(obj, k = 1, init = NA)
```

Arguments

obj	A factor in data.
init	A vector of initial values (correlation parameters followed by variance parameters) with an optional names attribute from the set {P, U, F} specifying the boundary constraint as positive, unconstrained or fixed, respectively.
b	Number of (sub-diagonal) bands in banded correlation models (default = 1).
k	Order of the model (for chol, ante) or number of factors (for sfa, facv, fa, rr) (default = 1).

Functions

- asr_cor(): Simple correlation model; correlation form.
- asr_corv(): Simple correlation model; homogeneous variance form.
- asr_corh(): Simple correlation model; heterogeneous variance form.
- asr_corb(): Banded correlation model with b bands; correlation form.
- asr_corbv(): Banded correlation model with b bands; homogeneous variance form.
- asr_corbh(): Banded correlation model with b bands, heterogeneous variance form.
- asr_corg(): General correlation model; correlation form.
- asr_corgv(): General correlation model; homogeneous variance form.
- asr_corgh(): General correlation model; heterogeneous variance form.
- asr_diag(): Diagonal variance model.
- asr_us(): Unstructured variance model.
- asr_chol(): Cholesky variance model of order k.
- asr_cholc(): Cholesky C variance model of order k.
- asr_ante(): Antedependence variance model of order k.
- asr_sfa(): Factor analytic model with k factors; the variance-covariance matrix is modelled on the correlation scale.
- asr_facv(): Factor analytic model with k factors; the variance-covariance matrix is modelled on the covariance scale.
- asr_fa(): Factor analytic model with k factors; sparse formulation where k "extra" levels are inserted in the mixed model equations.
- asr_rr(): Factor analytic model with k factors; reduced rank formulation of fa() where the default boundary constraints for the specific variances are set to F (fixed).

grass *Plant height disease study (multivariate form)*

Description

Data corresponds to plant height measurements on 14 plants on 5 occasions. The 14 plants were either diseased or healthy and were arranged in a glasshouse in a completely randomized design. Plant heights were measured 1, 3, 5, 7 and 10 weeks after the plants were placed in the glasshouse. There were 7 plants in each treatment.

Usage

```
grass
```

Format

A data frame with 7 columns and 14 rows:

- Tmt: Treatment, factor with 2 levels (diseased or healthy)
- Plant: Plant number ID, factor with 14 levels
- y1: Plant height at week 1
- y3: Plant height at week 3
- y5: Plant height at week 5
- y7: Plant height at week 7
- y10: Plant height at week 10

Source

J. Lamptey, Rothamsted Experimental Station, UK.

grassUV *Plant height disease study (univariate form)*

Description

Data corresponds to plant height measurements on 14 plants on 5 occasions. The 14 plants were either diseased or healthy and were arranged in a glasshouse in a completely randomized design. Plant heights were measured 1, 3, 5, 7 and 10 weeks after the plants were placed in the glasshouse. There were 7 plants in each treatment.

Usage

```
grassUV
```

Format

A data frame with 5 columns and 70 rows:

- Tmt: Treatment, factor with 2 levels (diseased or healthy)
- Plant: Plant number ID, factor with 14 levels
- Time: Time of measurement, factor with 5 levels
- HeightID: Trait of measurement height, factor with 5 levels
- y: Plant height

Source

J. Lamptey, Rothamsted Experimental Station, UK.

---

harvey *Weights of Hereford cattle*

Description

Average daily gain for 65 Hereford steers. The age at weaning, initial weight at the start of the test feeding period and average daily gain were recorded on 65 steers from 9 sires and 3 breeding lines; all of the steers were fed for the same length of time.

Usage

```
harvey
```

Format

A data frame with 8 columns and 65 rows:

- Calf: Calf number, factor with 65 levels
- Sire: Sire ID, factor with 9 levels
- Dam: Dam ID, factor with 1 level
- Line: Line number, factor with 3 levels
- ageOfDam: Age of Dam, integer value
- y1: Age at weaning
- y2: Initial weight
- y3: Average daily gain ($\times 100$)

References

W R Harvey (1960). "Least-squares Analysis of Data with Unequal Subclass Frequencies." Technical Report ARS 20-8, USDA, Agricultural Research Service. Reprinted with corrections as ARS H-4, 1975.

harvey.ped *Pedigree of Hereford cattle*

Description

Crossing history for 65 Hereford steers. Calf, sire and dam identities for 65 Hereford steers; the first three columns of data frame harvey.

Usage

```
harvey.ped
```

Format

A data frame with 3 columns and 65 rows:

- Calf: Calf number, integer value
- Sire: Sire ID, a character vector
- Dam: Dam ID, integer value

References

W R Harvey (1960). "Least-squares Analysis of Data with Unequal Subclass Frequencies." Technical Report ARS 20-8, USDA, Agricultural Research Service. Reprinted with corrections as ARS H-4, 1975.

harveyg.ped *Pedigree of Hereford cattle (with genetic groups)*

Description

Crossing history for 65 Hereford steers with genetic groups. Calf, sire and dam identities for 65 Hereford steers in 3 genetic groups. Additional rows to harvey.ped describe the genetic group structure.

Usage

```
harveyg.ped
```

Format

A data frame with 3 columns and 77 rows:

- Calf: Calf number, integer value
- Sire: Sire ID, a character vector
- Dam: Dam ID, integer value

References

W R Harvey (1960). "Least-squares Analysis of Data with Unequal Subclass Frequencies." Technical Report ARS 20-8, USDA, Agricultural Research Service. Reprinted with corrections as ARS H-4, 1975.

See Also

harvey harvey.ped

---

indep_vs *Default identity variance models*

Description

Model functions for identity variance structures.

Usage

```
id(obj)

idv(obj, init = NA)

idh(obj, init = NA)
```

Arguments

obj	A factor in data.
init	Optional vector of initial values with an optional names attribute from the set "P", "U", and "F", specifying the boundary constraint for each parameter as positive, unconstrained or fixed, respectively.

Details

The class of identity models includes the *null* correlation model id, and its homogeneous and heterogeneous variance forms idv and idh, respectively.

Functions

- asr_id(): Identity correlation model (no variance associated).
- asr_idv(): Identity variance model.
- asr_idh(): Heterogeneous identity (diagonal) variance model.

is.gamma *Checks if fitted model is in gamma scale*

Description

This internal function is used in summary.asreml and vpredict to verify if the parameters and the inverse of the average information matrix are in gamma scale.

Usage

```
is.gamma(object)
```

Arguments

object	An asreml object.

Value

A Boolean value identifying if the model is in gamma scale. The where.sigma2 attribure tells where the index of the sigma2 in vparameters object of the asreml list object.

Examples

```
## Not run:
asr.oats <- asreml(
  fixed = yield ~ Variety * Nitrogen,
  random = ~ Blocks / Wplots,
  data = oats)

is.gamma(asr.oats)

## End(Not run)
```

known_vs *Known variance structures*

Description

Model function associating a known variance structure with a factor in the data.

Usage

```
vm(obj, source, singG = NULL)

ide(obj)

ric(obj, source, singG = NULL)
```

Arguments

obj	A factor in data.
source	The known inverse or relationship matrix with the following descriptions: • 1. A sparse inverse variance matrix held in three-column coordinate form in row-major order. This triplet matrix must have class ginv from a call to ainverse(), or have attribute INVERSE set to TRUE. For backwards compatibility, a three-column data frame is also accepted. In either case, the source must have a rowNames attribute, or • 2. A sparse relationship matrix held in three-column coordinate form (as a matrix) in row-major order. If the attribute INVERSE is not set then FALSE is assumed; a rowNames attribute must be set, or • 3. A matrix (or Matrix object) with a dimnames attribute giving the levels of the model term being defined. This may be a relationship matrix or its inverse; if an inverse, it must have an INVERSE attribute set to TRUE, or • 4. A numeric vector of the lower triangular elements in row-major order. The vector must have a rowNames attribute, and if an inverse, it must have an INVERSE attribute set to TRUE.
singG	Ignored if source has class ginv or attribute INVERSE = TRUE; in such cases source must be one of: • 1. A sparse matrix in coordinate form with class ginv, or attribute INVERSE = TRUE, or • 2. An object of class matrix or Matrix with INVERSE = TRUE, or • 3. A vector assumed to be the lower triangle in row-major order with attribute INVERSE = TRUE. If source does not have class ginv, or the attribute INVERSE is FALSE or not set, and singG = NULL, then source is assumed a positive definite relationship matrix and singG is reset to "PD". Otherwise, a character string giving the state of the (to be inverted) source object: • "PD": Positive definite (default). • "ND": source is non-singular indefinite (positive and negative roots). In this case asreml ignores the indefinite condition and proceeds. • "PSD": source is positive semi-definite. In this case, asreml proceeds using Lagrange multipliers to process the matrix. Two cases arise: whether the singularity arises because an effect has zero variance or whether it arises as a linear dependence. An example of the first is when the genomic relationship matrix represents a dominance matrix, and the list of genotypes includes fully inbred individuals (which by definition have no dominance). An example of the second is when the list of genotypes includes clones. • "NSD": source is singular indefinite (positive, zero and negative roots). The indefinite condition is ignored and asreml proceeds using Lagrange multipliers as for "PSD" matrices.

Details

If source is of class Matrix, asreml will convert source internally to either a sparse triplet form (class dsparseMatrix), or dense vector form (class ddenseMatrix) for processing.

Functions

- `asr_vm()`: Create a model term associating a known relationship structure in `source` with a factor in `data`.
- `asr_ide()`: Create a term with the levels of `vm`, and modelled by the homogeneous form of the identity variance structure. The `vm` term must precede `ide` in the model for the factor levels to be found.
- `asr_ric()`: Create a model term associating a known relationship structure in `source` and residual additive genetic effects with a factor in `data`.

`lamb` *Footrot counts on lambs*

Description

Counts of two foot shape classes on 2513 lambs. The feet of 2513 lambs born in 1980 and 1981 from 5 mating groups were scored in two foot shape classes: 1) all four feet are normal, and 2) one foot is deformed. Two indicator variables were also recorded for the presence of the disease conditions scald and rot. The data are grouped into 68 sex, sire and mating group combinations.

Usage

```
lamb
```

Format

A data frame with 12 columns and 68 rows:

- cyr: Cross year, an integer with values 1 and 2
- Grp: Group, factor with 5 levels
- Sex: Sex, factor with 2 levels
- Sire: Sire number, factor with 18 levels
- xxx: A numeric vector
- tot: Binomial totals for each sex-sire-group combination
- l5: Incidence of foot shape class 2 (count)
- l4: Incidence of foot shape class 1 (count)
- ls: Incidence of scald disease (count)
- lr: Incidence of rot disease (count)
- prop: Foot shape class 2 (`l5`) as a proportion
- fail: Foot shape class 2 *failure* counts (`1-l5`)

References

A R Gilmour, R D Anderson, A L Rae (1985). "The Analysis of Binomial Data by a Generalized Linear Mixed Model." *Biometrika*, **72**, 593-599.

Levels *Combine factor levels*

Description

Function that forms a new model term from an existing factor by merging a subset of its levels. See the `combine` argument to `asreml`.

Usage

```
Levels(f, x)
```

Arguments

`f`	A factor in the data.
`x`	A vector of length `length(levels(f))` defining the levels of `f` to merge.

lrt *REML likelihood ratio test (REMLRT)*

Description

This function extracts the REML log-likelihood values and numbers of variance parameters from a sequence of `asreml` objects to compute a sequence of pairwise REML likelihood ratio tests. It uses the asymptotic chi-square distribution of the REMLRT statistic. The available method is for `asreml` class objects.

Usage

```
lrt(...)
```

Arguments

`...`	A sequence of `asreml` objects separated by comma.

Details

The REMLRT is only valid if the fixed effects are the same for both models. This requires not only the same fixed effects model, but also the same parameterisation. The models are arranged in increasing order of number of variance parameters and they are assumed to be nested in this sequence.

If the reduced model is obtained by setting positively-constrained variance parameters in the full model to zero, set `boundary = TRUE`. In this case the probability is computed using a mixture of chi-square distributions as described in Self and Liang (1987).

See Also

lrt.asreml

---

lrt.asreml *REML likelihood ratio test (REMLRT)*

Description

This function extracts the REML log-likelihood values and numbers of variance parameters from a sequence of asreml objects to compute a sequence of pairwise REML likelihood ratio tests. It uses the asymptotic chi-square distribution of the REMLRT statistic.

Usage

```
## S3 method for class 'asreml'
lrt(..., boundary = TRUE)
```

Arguments

...	A sequence of asreml objects assumed nested when arranged in increasing order of number of parameters (variance components).
boundary	If TRUE, hypothesized parameter values being tested lie on the boundary of their parameter space (default = TRUE).

Details

The REMLRT is only valid if the fixed effects are the same for both models. This requires not only the same fixed effects model, but also the same parameterisation. The models are arranged in increasing order of number of variance parameters and they are assumed to be nested in this sequence.

If the reduced model is obtained by setting positively-constrained variance parameters in the full model to zero, set boundary = TRUE. In this case the probability is computed using a mixture of chi-square distributions as described in Self and Liang (1987).

Value

A data frame with a row for each successive pairwise test, and columns for the REML likelihood ratio statistic, degrees of freedom and number of parameters.

References

S C Self, K Y Liang (1987). "Asymptotic Properties of Maximum Likelihood Estimators and Likelihood Ratio Tests Under Non-standard Conditions." *Journal of the American Statistical Association*, **82**, 605-610.

`matern_vs` *Matern variance structure*

Description

Model function for an extended Matern class The `mtrn` special function implements an extended Matern class (structure) which accommodates geometric anisotropy and a choice of metrics for random fields observed in two dimensions (*Haskard et al., 2007*). See the reference manual for further details.

Usage

```
mtrn(x, y, phi = NA, nu = 0.5, delta = 1.0, alpha = 0.0, lambda = 2)

mtrnv(x, y, phi = NA, nu = 0.5, delta = 1.0, alpha = 0.0, lambda = 2, init = 0.1)

mtrnh(x, y, phi = NA, nu = 0.5, delta = 1.0, alpha = 0.0, lambda = 2, init = 0.1)
```

Arguments

`x`	An object in `data` containing the *x* coordinates.
`y`	An object in `data` containing the *y* coordinates.
`phi`	The range parameter (default = `NA`).
`nu`	The smoothness parameter (default = `0.5`).
`delta`	Governs geometric anisotropy (default = `1.0`).
`alpha`	Governs geometric anisotropy (default = `0.0`).
`lambda`	Specifies the choice of metric. Options are `1` for city block, and `2` for Euclidean distance (default = `2`).
`init`	An optional vector of initial values for any variance parameters, with an optional `names` attribute from the set {`P`, `U`, `F`} specifying the boundary constraint as positive, unconstrained or fixed, respectively.

Details

If an argument to `mtrn` is numeric, it is treated as a starting value for estimation and given the constraint code `P` (positive). This behaviour can be altered by concatenating the numeric value followed by the constraint code ("P", "U" or "F") into a character string. If an argument is absent from the call, the corresponding parameter is held fixed at its default value.

Functions

- `asr_mtrn()`: Matern variance model; correlation form.
- `asr_mtrnv()`: Matern variance model; homogeneous variance form.
- `asr_mtrnh()`: Matern variance model; heterogeneous variance form.

References

K A Haskard, B R Cullis, A P Verbyla (2007). "Anisotropic Matern correlation and spatial prediction using REML." *Journal of Agricultural and Biological Sciences*, **12**, 147-160.

`meff` *Calculate marker effects*

Description

Function that estimates genetic regressor (marker) effects for a fitted `asreml` model object using a provided matrix of marker scores and an `asreml` fit with the specified relationship matrix. The available method is for `asreml` class objects.

Usage

```
meff(object, ...)
```

Arguments

`object`	An object of class `asreml`.
`...`	Arguments to `mef.asreml`

See Also

`meff.asreml`

`meff.asreml` *Calculate marker effects*

Description

Function that estimates genetic regressor (marker) effects for a fitted `asreml` model object using a provided matrix of marker scores and an `asreml` fit with the specified relationship matrix.

Usage

```
## S3 method for class 'asreml'
meff(object, effects = ~NULL, mef = list(), se = FALSE, evaluate = TRUE, ...)
```

Arguments

object	An asreml object.
effects	A one-sided formula giving the terms in the model (separated by "+") for which marker effects are to be calculated. If ~NULL, the term associated with the first variance matrix in the mef list will be used (default = ~NULL).
mef	A list associating a known relationship matrix (rm) used in the model with a matrix of regressor scores, with components in the form: rm = "regressor-scores".
se	If TRUE, calculates the standard errors of the effects (default = FALSE).
evaluate	If TRUE, evaluates the effects by a call to update.asreml; otherwise, it returns the unevaluated call (default = TRUE).
...	Additional arguments to asreml.

Value

An asreml object with a component mef, a list with length(mef) components containing the matrices of regressor effects and (optional) standard errors.

---

met *Multi-environmental wheat data from 8 locations with spatial coordinates*

Description

Multi-environmental wheat data from 8 locations with spatial coordinates

Usage

```
met
```

Format

A data frame with 8 columns and 2700 rows:

location: Factor with 8 levels representing the sites

rep: Factor with 2 levels of replication

ibk: Factor with 10 levels of incomplete blocks

check: Factor with 2 levels, 0 for test treatments, 1 for checks

gen: Factor with 273 levels of genotypes

col: Factor with 30 levels of columns

row: Factor with 71 levels of rows

yield: Recorded yield in bushels/acre

There are a few modifications from the original data: 1) three locations without phenotypic information have been removed (Mead, Kansas, and Westbred); 2) the check variable (originally called Entryc) has been changed to 0 (test) and 1 (checks); and 3) variable new was removed.

References

V Belamkar, M J Guttieri, W Hussain, D Jarquin, I El-basyoni, J Poland, A J Lorenz, P S Baenziger (2018). "Genomic Selection in Preliminary Yield Trials in a Winter Wheat Breeding Program." *G3 Genes|Genomes|Genetics*, **8**(8), 2735-2747.

`metric-1D_vs` *Metric-based variance models in one dimension.*

Description

Metric-based variance structures in one dimension. Includes one-dimensional exponential, gaussian and linear variance power models (`exp`, `gau`, `lvr`).

Usage

```
exp(x, init = NA, dist = NA)

expv(x, init = NA, dist = NA)

exph(x, init = NA, dist = NA)

gau(x, init = NA, dist = NA)

gauv(x, init = NA, dist = NA)

gauh(x, init = NA, dist = NA)

lvr(x, init = NA, dist = NA)

lvrv(x, init = NA, dist = NA)

lvrh(x, init = NA, dist = NA)
```

Arguments

`x`	An object in `data`.
`init`	An optional vector of initial values (power parameters followed by variance parameters) with an optional `names` attribute from the set {`P`, `U`, `F`} specifying the boundary constraint as positive, unconstrained or fixed, respectively.
`dist`	Optional numeric vector of coordinates (distances). If missing then the distances are obtained as `unique(obj)`.

Functions

- asr_exp(): Exponential (or power) model; correlation form.
- asr_expv(): Exponential (or power) model; homogeneous variance form.
- asr_exph(): Exponential (or power) model; heterogeneous variance form.
- asr_gau(): Gaussian power model; correlation form.
- asr_gauv(): Gaussian power model; homogeneous variance form.
- asr_gauh(): Gaussian power model; heterogeneous variance form.
- asr_lvr(): Linear variance model; correlation form.
- asr_lvrv(): Linear variance model; homogeneous variance form.
- asr_lvrh(): Linear variance model; heterogeneous variance form.

---

metric-2D_vs *Metric-based variance models in two dimensions*

Description

Metric-based variance structures in two dimensions. Includes two-dimensional isotropic exponential, gaussian, euclidean, linear variance, spherical and circular power models (iexp, igau, ieuc, ilv, sph, cir), anisotropic exponential and gaussian models (aexp, agau) and the Matern class (mtrn).

Usage

```
iexp(x, y, init = NA)

iexpv(x, y, init = NA)

iexph(x, y, init = NA)

aexp(x, y, init = NA)

aexpv(x, y, init = NA)

aexph(x, y, init = NA)

igau(x, y, init = NA)

igauv(x, y, init = NA)

igauh(x, y, init = NA)

agau(x, y, init = NA)

agauv(x, y, init = NA)
```

```
agauh(x, y, init = NA)

ieuc(x, y, init = NA)

ieucv(x, y, init = NA)

ieuch(x, y, init = NA)

ilv(x, y, init = NA)

ilvv(x, y, init = NA)

ilvh(x, y, init = NA)

sph(x, y, init = NA)

sphv(x, y, init = NA)

sphh(x, y, init = NA)

cir(x, y, init = NA)

cirv(x, y, init = NA)

cirh(x, y, init = NA)
```

Arguments

x	An object in data containing the *x* coordinates.
y	An object in data containing the *y* coordinates.
init	An optional vector of initial values (power parameters followed by variance parameters) with an optional names attribute from the set {P, U, F} specifying the boundary constraint as positive, unconstrained or fixed, respectively.

Functions

- asr_iexp(): Exponential (or power) model; correlation form.
- asr_iexpv(): Exponential (or power) model; homogeneous variance form.
- asr_iexph(): Exponential (or power) model; heterogeneous variance form.
- asr_aexp(): Anisotropic exponential variance model; correlation form.
- asr_aexpv(): Anisotropic exponential variance model; homogeneous variance form.
- asr_aexph(): Anisotropic exponential variance model; heterogeneous variance form.
- asr_igau(): Isotropic Gaussian variance model; correlation form.
- asr_igauv(): Isotropic Gaussian variance model; homogeneous variance form.
- asr_igauh(): Isotropic Gaussian variance model; heterogeneous variance form.

- `asr_agau()`: Anisotropic Gaussian variance model; correlation form.
- `asr_agauv()`: Anisotropic Gaussian variance model; homogeneous variance form.
- `asr_agauh()`: Anisotropic Gaussian variance model; heterogeneous variance form.
- `asr_ieuc()`: Isotropic Euclidean variance model; correlation form.
- `asr_ieucv()`: Isotropic Euclidean variance model; homogeneous variance form.
- `asr_ieuch()`: Isotropic Euclidean variance model; heterogeneous variance form.
- `asr_ilv()`: Isotropic linear variance model; correlation form.
- `asr_ilvv()`: Isotropic linear variance model; homogeneous variance form.
- `asr_ilvh()`: Isotropic linear variance model; heterogeneous variance form.
- `asr_sph()`: Spherical variance model; correlation form.
- `asr_sphv()`: Spherical variance model; homogeneous variance form.
- `asr_sphh()`: Spherical variance model; heterogeneous variance form.
- `asr_cir()`: Circular variance model; correlation form.
- `asr_cirv()`: Circular variance model; homogeneous variance form.
- `asr_cirh()`: Circular variance model; heterogeneous variance form.

`model_functions` *Model term constructor functions*

Description

This class of special functions constructs model terms with specific properties to be used with `asreml` syntax.

Usage

```
con(obj)

lin(obj)

pow(obj, p = 1, offset = 0)

pol(obj, t = 1, init = NA)

leg(obj,t = 1, init = NA)

spl(obj, k = 50, init = NA)

dev(obj, init = NA)

ma(obj)

at(obj, lvls)
```

```
and(obj, times = 1)

mbf(obj)

grp(obj)

dsum(model, levels = NULL, outer = FALSE)

C(obj, contr)
```

Arguments

obj	An object in the data frame. • mbf: A component name from the mbf list argument. • grp: A component name from the group list argument.
p	The exponent in a power function term (pow) (default = 1).
offset	Constant added to obj (default = 0).
t	A value. • pol: The maximum degree of a set of orthogonal polynomials formed from obj. If negative, the intercept in the polynomial is omitted (default = 1). • leg: The maximum degree of a set of Legendre polynomials formed from obj. If negative, the intercept is omitted (default = 1).
k	The number of equally-spaced knot points for a cubic smoothing spline. If zero or omitted, k is set to asreml.options()$knots (default 50).
init	Optional initial value for the default identity variance model idv when used in the random formula.
lvls	Vector of levels of the conditioning factor (obj) that define the conditioning covariates formed by at. If numeric, lvls indexes the levels vector of obj; that is levels(obj)[lvls].
times	It multiplies (may be non-integer) of the design matrix for obj are added to the preceding design matrix (default = 1).
model	A formula of the form ~A+B+...|Z, where A and B define variance matrices for simple or compound model terms, and Z is a simple conditioning factor whose levels identify and determine the number of sub-matrices in the direct sum. The "|" operator is applied associatively and operates with all terms on its left; that is, A+B|C implies (A+B)|C and is equivalent to A|C+B|C.
levels	A list of the length of the number of terms in the left-hand side of model that are separated by "+". The components of levels are vectors of unique values of Z. If there is only one term in the left-hand side of model (or if the context allows, see examples) then levels may be a vector. If levels is numeric, then these are ordinal numbers that refer to the sections defined by Z in unique(Z) order. If NULL then unique(Z) is used (default = NULL).
contr	An integer vector of contrast coefficients parallel to levels(obj).

Functions

- `asr_con()`: Sum to zero constraints.
- `asr_lin()`: Create a variate from `obj` (usually factors).
- `asr_pow()`: Create the model term: `(obj+offset)`^{p}.
- `asr_pol()`: Orthogonal polynomials.
- `asr_leg()`: Legendre polynomials.
- `asr_spl()`: Cubic smoothing spline (random component).
- `asr_dev()`: Spline deviations; create a factor from the variate `obj`.
- `asr_ma()`: Construct a term with a moving-average of order 1 design matrix from `obj`.
- `asr_at()`: Form a conditioning covariate from `obj` for each level of `obj` specified in the `lvls` argument.
- `asr_and()`: Multiply the design matrix for `obj` by `times` and add it to the preceding design matrix.
- `asr_mbf()`: Create a model term from covariates not stored in `data`.
- `asr_grp()`: Create a model term from covariates held in specific columns of `data`.
- `asr_dsum()`: Direct sum structures for residual models.
- `asr_C()`: Defines specific contrasts for a factor.

Examples

```
## Not run:

## Separable autoregressive residual model at each level of Site
residual = ~dsum(~ar1(Column):ar1(Row) | Site)

## Different residual models at different levels of Site.
## The ordinals in the levels list refer to the unique values of
## Site in unique(Site) order.
residual = ~dsum(~ar1(Column):ar1(Row) + id(Column):ar1(Row) | Site,
                 levels = list(c(1, 3), c(2)))

## Equivalent
residual = ~dsum(~ar1(Column):ar1(Row) | Site, levels = c("Site1", "Site3")) +
            dsum(~id(Column):ar1(Row)  | Site, levels = c("Site2"))

## "biological" Date (as factor) within Plot
residual = ~dsum(~ar1(Date) | Plot)

## "explicit" times (Date as variate)
residual = ~dsum(~exp(Date) | Plot)

## End(Not run)
```

na.method *Missing value action for* asreml *data frames*

Description

Function to deal with missing values in the data argument to asreml for the response and/or covariates. The options are: "include" to retain NAs in the data, "omit" to drop records with NAs, and "fail" to raise an exception if NAs are present.

Usage

```
na.method(y = c("include", "omit", "fail"), x = c("fail", "include", "omit"))
```

Arguments

y	Action to take if there are missing values in the response (default = "include").
x	Action to take if missing values are present in covariates (default = "fail").

Value

A list with components x and y.

naf *Barley study (naf)*

Description

Barley study (naf)

Usage

```
naf
```

Format

An object of class data.frame with 168 rows and 14 columns.

naf31H *Marker scores barley study (naf)*

Description

Marker scores barley study (naf)

Usage

```
naf31H
```

Format

An object of class `data.frame` with 118 rows and 19 columns.

naf32H *Marker scores barley study (naf)*

Description

Marker scores barley study (naf)

Usage

```
naf32H
```

Format

An object of class `data.frame` with 118 rows and 54 columns.

nassau *Loblolly pine clone trial*

Description

Total height of Loblolly pine trees at age 6 from field trial at Nassau, Florida, USA.

Usage

```
nassau
```

Format

A data frame with 5 columns and 6795 rows:

- Rep: Replicate, factor with 8 levels
- IncBlock: Incomplete block, factor with 80 levels
- CultureID: Culture treatment, factor with 2 levels
- clonefv: Clone ID, factor with 860 levels
- ht6: Tree height at age 6

References

M F Resende, P Munoz, M D Resende, D J Garrick, R L Fernando, J M Davis, E J Jokela, T A Martin, G F Peter, M Kirst (2012). "Accuracy of Genomic Selection Methods in a Standard Data Set of Loblolly Pine (Pinus taeda L.)." *Genetics*, **190**, 1503-1510.

---

nassau.grm *Loblolly genetic relationship matrix (GRM)*

Description

Genetic relationship matrix for 963 Loblolly pine clones derived from 4854 SNP markers.

Usage

```
nassau.grm
```

Format

A numeric matrix with 963 columns and 963 rows with attributes `"dimnames"` and `"scale"`. The relationship matrix is non-positive definite.

See Also

`nassau` `nassau.snp`

nassau.snp *Loblolly genetic marker scores*

Description

Genetic marker matrix for 963 Loblolly pine clones and 4854 SNP markers (coded as -1, 0, 1, and NAs).

Usage

```
nassau.snp
```

Format

A numeric matrix with 4854 columns and 963 rows with a "dimnames" attribute.

See Also

nassau nassau.grm

nin89 *Wheat advanced breeding trial*

Description

An advanced Nebraska Intrastate Nursery (NIN) breeding trial conducted at Alliance in 1988/89. Four replicates of 19 released cultivars, 35 experimental wheat lines and 2 additional triticale lines were laid out in a 22 row by 11 column rectangular array of plots. The varieties were allocated to the plots using a randomized complete block (RCB) design.

Usage

```
nin89
```

Format

A data frame with 11 columns and 242 rows:

- Variety: Variety name, factor with 56 levels
- Id: Variety ID, factor with 56 integral levels
- pid: Plot number, numeric variate
- raw: Plot weights
- Rep: Replicate, factor with 4 levels
- nloc: Trial location (= 4)

- yield: Grain yield (in t/ha)
- lat: Latitude, spatial coordinate
- long: Longitud, spatial coordinate
- Row: Field row, factor with 22 levels
- Column: Field column, factor with 11 levels

References

W W Stroup, P S Baenziger, D K Mulitze (1994). "Removing Spatial Variation from Wheat Yield Trials: A Comparison of Methods." *Crop Science*, **86**, 62-66.

`oats` *Oats variety-nitrogen factorial study*

Description

Yield of oats with four fertilizer treatments. The yield of oats from a split-plot field trial with 3 varieties and 4 levels of nitrogen fertilizer. The experiment was a split-plot with 6 blocks of 3 main plots (varieties), each split into 4 sub-plots (nitrogen).

Usage

```
oats
```

Format

A data frame with 9 columns and 72 rows:

- Blocks: Complete field replicates, factor with 6 levels
- Nitrogen: Nitrogen application, factor with 4 levels
- Subplots: Sub-plot within whole-plots; factor with 4 levels
- Variety: Variety names, factor with 3 levels
- Wplots: Whole-plots; factor with 3 levels
- yield: Grain yield (in 1/4 lbs.)
- Column: Field column index factor
- Row: Field row index factor
- nrate: Numeric nitrogen rates

References

F Yates (1935). "Complex Experiments." *Journal of the Royal Statistical Society Supplement*, **2**, 181-247.

orange *Orange tree circumference*

Description

Trunk circumferences (mm) of each of 5 trees recorded at 7 times; all trees were measured at the same time. Age of tree (in days since 31 December 1968) when measured and the corresponding season (spring or autumn) were also recorded.

Usage

```
orange
```

Format

A data frame with 4 columns and 35 rows:

- Tree: Tree ID, factor with 5 levels
- x: Age of tree
- circ: Trunk circumference (mm)
- Season: Season, factor with 2 levels (Spring and Autumn)

References

N R Draper, H Smith (1998). *Applied Regression Analysis*, 3rd edition. John Wyley and Sons, New York.

owt *Orange wether wool trial*

Description

To assess the importance of bloodline differences, a wether trial was conducted from 1984 to 1988 at Borenore near Orange. It involved 35 teams of wethers representing 27 bloodlines.

Usage

```
owt
```

Format

A data frame with 8 columns and 1485 rows:

- Tag: Tag ID, factor with 521 levels
- Site: Site ID, factor with 27 levels
- Bloodline: Tree ID, factor with 5 levels
- Team: Team ID, factor with 35 levels
- Year: Year, factor with 3 levels
- gfw: Greasy fleece weight (in kg)
- yield: Percentage of clean fleece weight to greasy fleece weight
- fdiam: Fibre diameter (in microns)

---

padVectorWithNA *Pad vector with* NAs

Description

Function to pad with NA.

Usage

```
padVectorWithNA(vec, lvec, indx)
```

Arguments

vec	Vector to pad.
lvec	Length of the vector after it needs to be padded.
indx	Indices in the padded vector associated with elements of vec, *i.e.* vec[i] goes into out[indx[i]].

Value

A padded vector

pixel *Repeated CT scans on dogs*

Description

This data set was collected by performing CT scans on dogs (dog) over time. Scans of the right- and left-side lymph nodes in the axillary region were recorded in form of pixel intensity. A total of 10 dogs were scanned over a period of 14 days. The dogs had received an intravenous application of a dye contrast at time zero.

Usage

```
pixel
```

Format

A data frame with 5 columns and 102 rows:

- rec: Record ID, number of record
- Dog: Dog ID, factor with 10 levels
- Side: Side, factor with 2 levels
- day: Day of post injection, numeric variate
- pixel: Pixel intensity, numeric response

References

J C Jose C. Pinheiro, D M Bates (2000). *Mixed-Effects Models in S and S-PLUS*. Springer, New York. doi:10.1007/b98882.

plot.asreml *Plot diagnostics for an* asreml *object*

Description

Four plots are generated: a histogram of the residuals, a Normal Q-Q plot, a plot of residuals against fitted values and a plot of residuals against unit number.

Usage

```
## S3 method for class 'asreml'
plot(
  x,
  res = "default",
  spatial = "trend",
  facet = TRUE,
  distribution = "norm",
  conf = 0.95,
  labels = FALSE,
  no_semi_transparency = FALSE,
  ...
)
```

Arguments

x	An asreml object.
res	The type of residuals to be plotted; see residuals.asreml (default = default).
spatial	If "plot" and an independent error has been fitted with units in the random formula, these are added to the residuals, otherwise if "trend" then units are not added even if present in the model (default = "trend").
facet	If TRUE, multi-panel conditioning plots are produced for models with multi-section residual structures (default = TRUE).
distribution	A character string specifying the distribution for Q-Q plots. Density and quantile functions for many standard probability distributions are available in the "stats" package. The distribution name is automatically prefixed with "d" and "q" (default = "norm").
conf	Value specifying confidence level (0 to 1) to be used to present confidence intervals (default = 0.95).
labels	If TRUE, points are labelled by unit number (default = FALSE).
no_semi_transparency	This should be set to TRUE if the device being used does not support semi-transparency (default = FALSE).
...	Additional arguments.

Details

If the residual structure of the model contains multiple sections, the default plots are conditioned on the factor whose levels define the sections; Normal Q-Q plots are only produced if facet = TRUE. For multivariate analyses, the plots are conditioned on trait.

Value

An invisible list of ggplot2 objects.

plot.varioGram *Plot an empirical variogram*

Description

A plot method for varioGram objects returned from a call to varioGram.asreml.

Usage

```
## S3 method for class 'varioGram'
plot(x, npanels = NA, scale = TRUE, ...)
```

Arguments

x	An asreml object.
npanels	The number of lattice panels. If NA, it is set to the number of groups in the object (default = NA).
scale	If TRUE and there are multiple groups in the object, the response in all groups is scaled relative to the maximum (default = TRUE).
...	Arguments to be passed to the lattice functions xyplot or wireframe.

Value

An invisible lattice object.

plot_coef *Plot model coefficients*

Description

Function to extract and generate a Q-Q plot of fitted coefficients from an asreml object.

Usage

```
plot_coef(object, ...)
```

Arguments

object	An object of class asreml.
...	Arguments to plot_coef.asreml.

See Also

plot_coef.asreml

plot_coef.asreml *Plot model coefficients*

Description

Function to extract and generate a Q-Q plot of fitted coefficients from an asreml object.

Usage

```
## S3 method for class 'asreml'
plot_coef(
  object,
  formula,
  distribution = "norm",
  conf = 0.95,
  labels = TRUE,
  no_semi_transparency = FALSE,
  ...
)
```

Arguments

object	An object of class asreml.
formula	A one-sided formula identifying the model term to plot.
distribution	A character string specifying the distribution for Q-Q plots. Density and quantile functions for many standard probability distributions are available in the "stats" package. The distribution name is automatically prefixed with "d" and "q" (default = "norm").
conf	Value specifying confidence level (0 to 1) to be used to present confidence intervals (default = 0.95).
labels	If TRUE, points are labelled from the dimnames attribute of the coefficients. If labels is a vector the provided elements are used to label points (default = TRUE).
no_semi_transparency	This should be set to TRUE if the device being used does not support semi-transparency (default = FALSE).
...	Additional arguments.

predict.asreml *Predict linear functions of model effects*

Description

An instance of the generic method predict for objects of class asreml. It forms a linear function of the vector of fixed and random effects in the linear model to obtain an estimated or predicted value.

Usage

```
## S3 method for class 'asreml'
predict(
  object = NULL,
  classify = character(0),
  levels = list(),
  present = character(0),
  ignore = character(0),
  use = character(0),
  except = character(0),
  only = character(0),
  associate = formula("~NULL"),
  average = list(),
  vcov = FALSE,
  sed = FALSE,
  parallel = FALSE,
  aliased = FALSE,
  design.points = list(),
  evaluate = TRUE,
  ...
)
```

Arguments

object	An asreml object.
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.
levels	A list, named by the margins of the classifying table, of vectors specifying the levels at which predictions are required. If omitted, factors are predicted at each level, simple covariates are predicted at their overall mean and covariates used as a basis for splines or orthogonal polynomials are predicted at their design points. Additional prediction points for spline terms should be included in the design matrix with the asreml knot.points argument and included in the predict set

with the predict design.points argument. The factors mv and units are always ignored.

present — A character vector specifying which variables to include in the present averaging set. The present set is used when averaging is to be based only on cells with data. The present set may include variables in the classify set but not those in the average set.

If a list, there can be a maximum of two components, each a character vector of variable names, representing non-overlapping present categorisations and one optional component named prwts containing a vector of weights to be used for averaging the **first** present table only. The vector(s) of names may include variables in the classify set but not those in the average set.

ignore — A character vector specifying which variables to ignore in forming the predictions.

use — A character vector specifying which variables to add to the prediction model after the default rules have been invoked.

except — A character vector specifying which variables to exclude in the prediction process. That is, the prediction model includes all fitted model terms not in the except list.

only — A character vector specifying which variables (only) form the prediction model, that is, the default rules are not invoked.

associate — A one-sided formula specifying terms in up to two independent nested hierarchies. The factors in each hierarchy are written as a compound term separated by the ":" operator and in *left-to-right* outer to inner nesting order. Nested hierarchies are separated by the "+" operator; only one "+" operator is currently permitted, giving a maximum of two associate lists.

average — A list, named by the margins of the classifying table, specifying which variables to include in the averaging set. Optionally, each component of the list is a vector specifying the weights to use in the averaging process. If omitted, equal weights are used.

vcov — If TRUE, the full variance-covariance matrix of the predicted values is returned in a component vcov (default = FALSE).

sed — If TRUE, the full standard error of difference (SED) matrix of the predicted values is returned in a component sed (default = FALSE).

parallel — If TRUE, the levels of the classify factors given in the levels list are expanded in parallel. In this case, levels must be specified for all factors in the classify set, and they must be of equal length (default = FALSE).

aliased — If TRUE, the predicted values are returned for non-estimable functions. However, this is rarely a satisfactory solution (default = FALSE).

design.points — A list with named components where each component is a list or matrix (for two dimensions), or vector (single dimension) of user-supplied prediction design points for: spl(), pol(), dev() or metric type models.

If an element of this list is a list of length two, then the first vector component is taken as the x coordinates and the second as the y coordinates. If a component is a matrix, then it is assumed that the (x, y) coordinates occupy columns 1 and 2, respectively. The names of design.points must match exactly those used in the model functions.

evaluate	If FALSE, an unevaluated call to update.asreml is returned, otherwise the call is evaluated. Setting FALSE returns a list that may be used with the predict argument in a call to asreml (default = TRUE).
...	Additional arguments to asreml.

Details

The prediction process forms a linear function of the vector of fixed and random effects in the linear model to obtain a predicted value for a quantity of interest. It is primarily used for predicting tables of adjusted means. If the table is based on a subset of the explanatory variables then the other variables need to be accounted for. It is usual to form a predicted value either at specified values of the remaining variables, or averaging over them in some way.

Prediction equations are formed just prior to the final iteration in asreml. The predict.asreml method passes the list of user specifications for the prediction design matrix to the REML routines through the predict argument of asreml. Predicted values and standard errors are returned in the predictions component of the asreml object. In forming the predictions, predict.asreml calls update.asreml to re-run the model from its previous solution.

Value

The full asreml object is not returned, only the predictions element containing the following components:

- pvals: A data frame of predicted values with class asreml.predict.
- sed: Optional matrix of class dspMatrix of standard errors of difference (SED).
- vcov: Optional variance-covariance matrix of class dspMatrix of the predicted values.
- avsed: Average of the standard error of differences (SED).

References

S J Welham, B R Cullis, B J Gogel, A R Gilmour, R Thompson (2004). "Prediction in linear mixed models." *Australian and New Zealand Journal of Statistics*, **46**, 325-347.

---

print.asreml.predict *Print predictions of linear functions*

Description

A print method for asreml.predict objects.

Usage

```
## S3 method for class 'asreml.predict'
print(x, digits = getOption("digits"), ...)
```

Arguments

x	An object of class asreml.predict from a call to predict.asreml.
digits	Numeric precision.
...	Additional arguments.

---

print.wald *Print a* wald *object*

Description

Print method similar to print.anova for objects with class wald.

Usage

```
## S3 method for class 'wald'
print(
  x,
  digits = max(getOption("digits") - 2L, 3L),
  signif.stars = getOption("show.signif.stars"),
  ...
)
```

Arguments

x	An object with class wald from a call to wald.asreml.
digits	Numeric precision.
signif.stars	Should stars be printed on summary tables of coefficients (default = getOption("show.signif.stars"))
...	Additional arguments to be passed to stats::printCoefmat.

---

rats *Reproductive study on rats*

Description

Litter weights of pups from rats given 3 doses (control, low and high) of an experimental compound affecting reproductive performance. Thirty female rats (dams) were randomly split into 3 groups of 10 and each group randomly assigned a dosing level. A total of 3 litters had to be dropped from the high dose level.

Usage

```
rats
```

Format

A data frame with 6 columns and 322 rows:

- Dose: Dose application, factor with 3 levels
- Sex: Sex; factor with 2 levels
- littersize: Number of individuals by litter, a covariate
- Dam: Name of the dam, factor with 27 levels
- Pup: Number of pup, factor with 18 levels
- weight: Individual pup weights (in grams)

References

A P Dempster, C M Patel, M R Selwyn, A J Roth (1984). "Statistical and Computational Aspects of Mixed Model Analysis." *Journal of the Royal Statist. Series C.*, **33**(2), 203-214.

`residuals.asreml` *Extract model residuals*

Description

Extracts residuals from `asreml` objects.

Usage

```
## S3 method for class 'asreml'
residuals(
  object,
 type = c("working", "deviance", "stdDeviance", "pearson", "response", "stdCond"),
  spatial = c("trend", "plot"),
  ...
)
```

Arguments

`object`	An `asreml` object.
`type`	Type of residuals, options are: `"deviance"`, `"pearson"`, `"working"`, `"response"`, `"response"` or `"stdCond"` (default = `"working"`).
`spatial`	If a second independent error term has been fitted by including `units` in the random formula, the residuals will have the `units` E-BLUPs added if `"plot"` (default = `"trend"`).
`...`	Additional arguments.

Value

A numeric vector containing the model residuals.

`rice` *Rice resistance to bloodworms (univariate form)*

Description

An investigation of the tolerance of rice varieties to attack by the larvae of bloodworms. The experiment commenced with the transplanting of rice seedlings into trays. Each tray contained a total of 32 seedlings and the trays were paired so that a control tray (no bloodworms) and a treated tray (bloodworms added) were grown in a controlled environment room for the duration of the experiment. After this, rice plants were carefully extracted, the root system washed and root area determined for the tray using an image analysis system. Two pairs of trays, each pair corresponding to a different variety, were included in each run. A new batch of bloodworm larvae was used for each run. A total of 44 varieties was investigated with 3 replicates of each. The variety concurrence was such that: 8 varieties occurred with only one other variety, 22 with two other varieties and the remaining 14 with three different varieties.

Usage

```
rice
```

Format

A data frame with 6 columns and 264 rows:

- Pair: Pair ID, factor with 132 levels
- rootwt: Root weight
- Run: Run, factor with 66 levels
- sqrtroot: Square root of `rootwt`
- Tmt: Treatment, factor with 2 levels (Control and Exposed)
- Variety: Variety name, factor with 44 levels

References

M M Stevens, K M Fox, G N Warren, B R Cullis, N E Coombes, L G Lewin (1999). "An Image Analysis Technique for Assessing Resistance in Rice Cultivars to Root-feeding Chironomid Midge Larvae (Diptera: Chironomidae)." *Field Crops Research*, **66**, 25-26.

riceMV *Rice resistance to bloodworms (multivariate form)*

Description

An investigation of the tolerance of rice varieties to attack by the larvae of bloodworms. Multivariate form of rice with root weight in two variates corresponding to the levels of the applied treatment (control or exposed to bloodworms).

Usage

```
riceMV
```

Format

A data frame with 7 columns and 132 rows:

- Pair: Pair ID, factor with 132 levels
- Run: Run, factor with 66 levels
- Variety: Variety name, factor with 44 levels
- yc: Root weight for the Control treatment
- ye: Root weight for the Exposed treatment
- syc: The square root of yc
- sye: The square root of ye

References

M M Stevens, K M Fox, G N Warren, B R Cullis, N E Coombes, L G Lewin (1999). "An Image Analysis Technique for Assessing Resistance in Rice Cultivars to Root-feeding Chironomid Midge Larvae (Diptera: Chironomidae)." *Field Crops Research*, **66**, 25-26.

See Also

rice

shf *Slate Hall farm wheat variety trial*

Description

Data from a field experiment to compare 25 varieties of wheat at Slate Hall farm, UK, 1976. The trial was a balanced lattice with 25 varieties in 6 replicates, arranged in a 15 column by 10 row grid of plots.

Usage

```
shf
```

Format

A data frame with 7 columns and 150 rows:

- Rep: Complete field replicates, factor with 6 levels
- RowBlk: Incomplete row blocks, factor with 30 levels
- ColBlk: Incomplete column blocks, factor with 30 levels
- Row: Long row blocks, factor with 10 levels
- Column: Long column blocks, factor with 15 levels
- Variety: Variety names, factor with 25 levels
- yield: Grain yield (in grams)

References

R A Kempton, P N Fox (1997). *Statistical Methods for Plant Variety Evaluation*. Chapman and Hall.

sp2mat *Convert sparse matrix to dense form*

Description

Function that converts a sparse matrix in three-column coordinate form to a dense matrix form.

Usage

```
sp2mat(x)
```

Arguments

x	A three-column matrix containing the row and column indices and the matrix element, respectively.

Details

If the sparse matrix inherits class `ginv`, the returned matrix preserves the A^{-1} attributes.

---

sp2Matrix *Convert sparse matrix to* `Matrix` *object*

Description

Function that converts a sparse matrix in three-column coordinate form to a `Matrix` object.

Usage

```
sp2Matrix(x, dense = FALSE, triplet = FALSE)
```

Arguments

`x`	A three-column matrix containing the row and column indices and the matrix element, respectively, of the sparse matrix.
`dense`	If `TRUE`, the result is stored as a dense symmetric `dspMatrix` object, otherwise a sparse symmetric matrix (`dsCMatrix` or `dsTMatrix`, depending on the choice of the `triple` argument) is returned (default = `FALSE`).
`triplet`	If `TRUE`, the result is a `dsTMatrix` object (triplet form), otherwise a `dsCMatrix` object is returned (default = `FALSE`).

Details

If the sparse matrix inherits class `ginv`, the returned object preserves the A^{-1} attributes.

---

splinek *Spline knot points*

Description

Return the spline knot points generated via the `spl()` model function.

Usage

```
splinek(x, k = 0, predict.points = NULL, core_version = NULL)
```

Arguments

x	A numeric vector.
k	The number of equally-spaced knot points for a cubic smoothing spline. If zero or omitted, k is set to asreml.options()$knots, in which case the number of knot points used will be: min(unique(x), asreml.options()$knots) (default = 50).
predict.points	An optional vector of points to be included in the design matrix for prediction.
core_version	The version of the algorithmic core to use, not usually required.

Value

A list including the following components:

- knotpoints: The knot points generated by asreml.
- zknots: The spline design matrix at the knot points.

Subset *Subset a factor*

Description

Function that forms a new model term from an existing factor by selecting a subset of its levels. See the prune argument to asreml.

Usage

```
Subset(f, x)
```

Arguments

f	A factor in the data.
x	A character or numeric vector of levels to select.

summary.asreml *Summarize an* asreml *object*

Description

A summary method for objects inheriting from class asreml.

Usage

```
## S3 method for class 'asreml'
summary(
  object,
  param = c("sigma", "gamma"),
  coef = FALSE,
  vparameters = FALSE,
  ...
)
```

Arguments

object	An asreml object.
param	If "sigma", variance components are reported on the *sigma* scale only; otherwise, if "gamma", an additional column of variance ratios is returned (default = "sigma").
coef	If TRUE, the coefficients and their standard errors are included in the returned object (default = FALSE).
vparameters	If TRUE, the variance component parameters are included in the returned object in list form (default = FALSE).
...	Additional arguments.

Value

A list of class summary.asreml with the following components:

- call: The call component from object.
- loglik: The log-likelihood value, loglik component, from object.
- nedf: The nedf component from object.
- sigma: The square of the residual variance, sqrt(object$sigma2).
- varcomp: A data frame summarizing the random parameter vector (object$vparameters). Variance component ratios are included if param = "gamma", and a measure of precision (standard error) is included along with boundary constraints at termination and the percentage change in the final iteration.
- aic: Akaike information criterion for the fitted model.
- bic: Bayesian information criterion for the fitted model.

- distribution: A character string identifying the error distribution(s) if object$deviance != 0.
- link: A character string identifying the link function(s) if object$deviance != 0.
- deviance: The deviance from the fit if object$deviance != 0.
- heterogeneity: Variance heterogeneity (calculated as deviance/nedf) if object$deviance != 0.
- coef.fixed: A matrix of coefficients (solutions) and their standard errors for fixed effects if coef = TRUE.
- coef.random: A matrix of coefficients (solutions) and their standard errors for random effects if coef = TRUE.
- coef.sparse: A matrix of coefficients (solutions) and their standard errors for fixed effects included in the sparse set if coef = TRUE.
- vparameters: A list of variance structures with matrices converted to full dense form. For ante and chol models the components are given in varcomp and returned in gammas in variance-covariance form.

---

switchVersion *Detach and reload the* asreml *package*

Description

Detach the asreml package and reload it with access to the specified cores

Usage

```
switchVersion(core = NULL, qt = TRUE)
```

Arguments

core	If not supplied, will default to the "next" version.
qt	If TRUE the switch will be (reasonably) quiet (default = TRUE).

---

time_series_vs *Time series type correlation and variance structures.*

Description

The class of time series type models includes autoregressive models of order 1, 2 and 3 (ar1, ar2 and ar3), symmetric autoregressive (sar), constrained autoregressive order 2 (sar2), moving average models of order 1 and 2 (ma1, ma2) and the autoregressive-moving average model (arma). All of these are available as a correlation structure form, or as a homogeneous or heterogeneous form.

Usage

```
ar1(obj, init = NA)

ar1v(obj, init = NA)

ar1h(obj, init = NA)

ar2(obj, init = NA)

ar2v(obj, init = NA)

ar2h(obj, init = NA)

ar3(obj, init = NA)

ar3v(obj, init = NA)

ar3h(obj, init = NA)

sar(obj, init = NA)

sarv(obj, init = NA)

sarh(obj, init = NA)

sar2(obj, init = NA)

sar2v(obj, init = NA)

sar2h(obj, init = NA)

ma1(obj, init = NA)

ma1v(obj, init = NA)

ma1h(obj, init = NA)

ma2(obj, init = NA)

ma2v(obj, init = NA)

ma2h(obj, init = NA)

arma(obj, init = NA)

armav(obj, init = NA)

armah(obj, init = NA)
```

Arguments

obj	A factor in data.
init	A vector of initial values (correlation parameters followed by variance parameters) with an optional names attribute from the set {P, U, F} specifying the boundary constraint as positive, unconstrained or fixed, respectively.

Functions

- asr_ar1(): Autoregressive model of order 1; correlation form.
- asr_ar1v(): Autoregressive model of order 1; homogeneous variance form.
- asr_ar1h(): Autoregressive model of order 1; heterogeneous variance form.
- asr_ar2(): Autoregressive model of order 2; correlation form.
- asr_ar2v(): Autoregressive model of order 2; homogeneous variance form.
- asr_ar2h(): Autoregressive model of order 2; heterogeneous variance form.
- asr_ar3(): Autoregressive model of order 3; correlation form.
- asr_ar3v(): Autoregressive model of order 3; homogeneous variance form.
- asr_ar3h(): Autoregressive model of order 3; heterogeneous variance form.
- asr_sar(): Symmetric autoregressive model; correlation form.
- asr_sarv(): Symmetric autoregressive model; homogeneous variance form.
- asr_sarh(): Symmetric autoregressive model; heterogeneous variance form.
- asr_sar2(): Constrained autoregressive model of order 3; correlation form.
- asr_sar2v(): Constrained autoregressive model of order 3; homogeneous variance form.
- asr_sar2h(): Constrained autoregressive model of order 3; heterogeneous variance form.
- asr_ma1(): Moving average model of order 1; correlation form.
- asr_ma1v(): Moving average model of order 1; homogeneous variance form.
- asr_ma1h(): Moving average model of order 1; heterogeneous variance form.
- asr_ma2(): Moving average model of order 2; correlation form.
- asr_ma2v(): Moving average model of order 2; homogeneous variance form.
- asr_ma2h(): Moving average model of order 2; heterogeneous variance form.
- asr_arma(): Autoregressive-moving average model; correlation form.
- asr_armav(): Autoregressive-moving average model; homogeneous variance form.
- asr_armah(): Autoregressive-moving average model; heterogeneous variance form.

tr *Plot trace convergence*

Description

Scatter plots of model component values against iteration for each random component for a fitted model. The available method is for `asreml` class objects.

Usage

```
tr(object, ...)
```

Arguments

`object`	An object of class `asreml`.
`...`	Arguments to `tr.asreml`

Details

Values are extracted from the `trace` matrix of an `asreml` fitted object. The (first) three rows of this matrix corresponding to `LogLik`, `Sigma2` and `DF`, respectively, are not plotted by default.#'

See Also

`tr.asreml`

tr.asreml *REML convergence trace for* `asreml` *objects.*

Description

Scatter plots of model component values against iteration for each random component for a fitted model.

Usage

```
## S3 method for class 'asreml'
tr(
  object,
  components = seq(4, nrow(object$trace)),
  iter = seq(1, ncol(object$trace)),
  ...
)
```

Arguments

`object`	An `asreml` object.
`components`	A numeric vector of row numbers of the `trace` matrix to include; default is all rows excluding rows 1, 2 and 3.
`iter`	A numeric vector of iteration numbers to include; default is all columns of `trace`.
`...`	Additional arguments.

Details

Values are extracted from the `trace` matrix of an `asreml` fitted object. The (first) three rows of this matrix corresponding to `LogLik`, `Sigma2` and `DF`, respectively, are not plotted by default.

Value

An invisible list of `ggplot2` objects.

`Units` *Unit level numbers*

Description

Function that forms a new model term from an existing factor by choosing a subset of its record numbers. See the `uid` argument to `asreml`.

Usage

```
Units(f, n = 0)
```

Arguments

`f`	A factor in the data.
`n`	A character or numeric scalar defining the records of `f` to select (default = `0`).

`update.asreml` *Update an* `asreml` *model fit*

Description

Extracts and evaluates the `call` from the fitted `asreml` object, replacing any arguments with changed values. In particular, `G.param` and `R.param` are automatically updated with those stored in the object. It is also used to perform a few additional iterations of a model that has not converged.

Usage

```
## S3 method for class 'asreml'
update(
  object,
  fixed.,
  random.,
  sparse.,
  residual.,
  keep.order = TRUE,
  evaluate = TRUE,
  ...
)
```

Arguments

`object`	A valid `asreml` object with a `call` component, the expression used to create itself.
`fixed.`	Changes to the fixed formula. This is a two-sided formula where `"."` is substituted for existing components in the `fixed` component of `object$call`.
`random.`	Changes to the random formula. This is a one-sided formula where `"."` is substituted for existing components in the right-hand side of the `random` component of `object$call`.
`sparse.`	Changes to the sparse formula. This is a one-sided formula where `"."` is substituted for existing components in the right-hand side of the `sparse` component of `object$call`.
`residual.`	Changes to the residual formula. This is a one-sided formula where `"."` is substituted for existing components in the right-hand side of the `residual` component of `object$call`.
`keep.order`	If `TRUE`, the ordering of terms is retained in the updated formulae (default = `TRUE`).
`evaluate`	If `TRUE`, the new call is evaluated; otherwise the call is returned as an unevaluated expression (default = `TRUE`).
`...`	Additional arguments to the call, or arguments with changed values.

Details

In addition to any other changes, update.asreml replaces the arguments R.param and G.param with object$R.param and object$G.param, respectively, creating a new fitted object when run using the parameter values from a previous model as initial values.

Value

Returns either a new updated asreml object, or an unevaluated expression for creating such an object.

Examples

```
## Not run:
oats.vsr <- asreml(yield ~ Variety+Nitrogen, random = ~ Blocks/Wplots, data = oats)
oats2.vsr <- update(oats.vsr, fixed = . ~ . + Variety:Nitrogen)

## End(Not run)
```

---

user_vs *User-defined variance models or structures*

Description

Specify an external function that provides a variance-covariance matrix, or its inverse.

Usage

```
own(obj, fun = "myowng", is.variance = TRUE)
```

Arguments

obj	A factor in data.
fun	The name (as a character string) of an R function to compute the variance matrix and its derivative(s). This function must accept a single argument: • order: a scalar giving the dimension of the structure being defined, and return a list containing the matrix (the variance matrix or its inverse) and the corresponding derivative matrices. The variance matrix may be a dense matrix class object, a vector being the lower triangle in row-major order, or a three-column matrix in coordinate form in row-major order. This object may have an attribute INVERSE, a logical scalar identifying the structure as a variance matrices or its inverse. If INVERSE attribute is absent the default is FALSE.
is.variance	If TRUE, then own assumes the resulting structure is a variance matrix, otherwise a correlation matrix is assumed. This only affects the default scaling factor (default = TRUE).

Details

The own variance model allows users to specify external variance structures. This requires the user to provide an R function that accepts a single argument, the leading dimension of the structure, and forms the variance matrix (or its inverse). The R function may invoke compiled code if necessary.

Examples

```
## Not run:
## An own ar1 function
ar1.asr <- asreml(yield ~ Variety,
                  residual = ~ar1(Row):own(Column, "ar1", 0.1, "R"),
                  data = shf)

## where the function "ar1" is defined as:

## ar1 <- function(order, kappa) {
##   t <- 1:order
##   H <- abs(outer(t, t, "-"))
##   V <- kappa^H
##   ## derivative
##   dV <- H*(kappa^(H-1))
##   return(list(V, dV))
## }

## End(Not run)
```

---

varioGram *Empirical variogram constructor method*

Description

Function that calculates the empirical variogram from an asreml object. The available method is for asreml class objects.

Usage

```
varioGram(object, ...)
```

Arguments

object	An object of class asreml.
...	Arguments to varioGram.asreml

Details

Calls asr_varioGram to calculate the empirical semi-variogram.

Value

A data frame including the following components:

- x: The original *x* coordinates.
- y: The original *y* coordinates.
- gamma: The variogram estimate.
- distance: The average distance for pairs in the lag.
- np: The number of pairs in the lag.
- angle: Direction, if not a regular grid.

See Also

varioGram.asreml

---

varioGram.asreml *Empirical variogram constructor method*

Description

Function that calculates the empirical variogram from an asreml object.

Usage

```
## S3 method for class 'asreml'
varioGram(
  object,
  type = "default",
  spatial = "trend",
  formula = ~NULL,
  composite = TRUE,
  model = c("empirical"),
  metric = c("euclidean", "manhattan"),
  angle = 0,
  angle.tol = 180,
  nlag = 20,
  maxdist = 0.5,
  xlag = NA,
  lag.tol = 0.5,
  grid = TRUE,
  ...
)
```

Arguments

object	An object of class asreml.
type	Type of residuals, see residuals.asreml for more details (default = "default").
spatial	Whether to include a nugget effect; see residuals.asreml for more details (default = "trend").
formula	An optional model formula designed to extract *residuals* from the *random* component of the model rather than the *residual* component. This is a two-sided formula where the response is a pattern in the style required by the pattern argument of coef.asreml.
composite	Argument to asr_varioGram. To be used for data on a regular grid. If TRUE, the average of the variograms in quadrants *(x, y)* and *(x, -y)* is returned. Otherwise, both variograms are returned and identified as quadrants 1 and 4 (default = TRUE).
model	Argument to asr_varioGram. At the present, it can only be "empirical".
metric	Argument to asr_varioGram. The distance between *(x, y)* points. Options are: "euclidean" or "manhattan" (default = "euclidean").
angle	Argument to asr_varioGram. A vector of directions. Angles are measured in degrees anticlockwise from the *x* axis (default = 0).
angle.tol	Argument to asr_varioGram. The angle subtended by each direction. That is, an arc angle +/- angle.tol/2 (default = 180, which gives an omnidirectional variogram).
nlag	Argument to asr_varioGram. The maximum number of lags (default = 20).
maxdist	Argument to asr_varioGram. The fraction of the maximum distance to include in the calculation. The default is half the maximum distance in the data: 0.5.
xlag	Argument to asr_varioGram. The width of the lags (default = NA).
lag.tol	Argument to asr_varioGram. The distance tolerance (default = 0.5).
grid	Argument to asr_varioGram. If FALSE, forces polar variograms if *(x, y)* specifies a regular grid (default = TRUE).
...	Additional arguments.

Details

Calls asr_varioGram to calculate the empirical semi-variogram.

Value

A data frame including the following components:

- x: The original *x* coordinates.
- y: The original *y* coordinates.
- gamma: The variogram estimate.
- distance: The average distance for pairs in the lag.
- np: The number of pairs in the lag.
- angle: Direction, if not a regular grid.

Examples

```
## Not run:
data(barley)
shf.vsr <- asreml(yield ~ Variety, residual = ~ ar1(Row):ar1(Column),
                  data = shf)
variogram(shf.vsr)

## End(Not run)
```

---

vcm.lm *Specify constraints among variance components*

Description

Construct a constraints matrix that specifies linear constraints among variance components.

Usage

```
vcm.lm(
  form,
  data,
  drop.unused.levels = TRUE,
  intercept = FALSE,
  na.action = na.fail
)
```

Arguments

form	A model formula including at least one factor with up to n_c levels, where n_c is the number of variance parameters to be considered in the constrained set.
data	A data frame with a factor Vparameter, whose levels are taken from the full set of variance parameters, in which to resolve the names in form.
drop.unused.levels	If TRUE, unused levels in factors are removed (default = TRUE).
intercept	if FALSE, the intercept is not included in the call to model.matrix when forming the constraints matrix (default = FALSE).
na.action	The default, na.fail, is to terminate abnormally if missing values are present.

Details

Variance parameter constraints are specified through a design matrix F from a simple linear model. Let κ be the r-vector of original variance parameters (for either the sigma or gamma parameterisation) from which we wish to specify linear relationships of the form $\kappa = F\kappa_n$, where κ_n is the c-vector of parameters in the new set. In the simple, case where the r parameters are constrained to be equal $c = 1$, the r original parameters are all equal to the one new parameter and F will contain a column of ones.

The matrix F is given as the value to the `vcm` argument of `asreml`. F must have a `dimnames` attribute with the names of κ as its row names.

A data frame containing a factor `Vparameter`, whose levels are the r names of the variance parameters is returned by `asreml` when `start.values = TRUE`. The matrix F is obtained from a call to `model.matrix` using `form` and additional factors derived from or interacting with `Vparameter`.

Value

An $r \times c$ matrix F specifying the variance parameter constraints where c is the length of the reduced vector of variance parameters. In a simple case with r parameters and the $r-1$ and r parameters are constrained to be equal, then $c = r - 1$ and the j^{th} $(1 = j < c)$ column of F has 1 in the j^{th} row and zero elsewhere; the c^{th} column has 1 in the $c = (r-1)$ and r rows and zero elsewhere.

Examples

```
# Suppose there are four variance parameters: g1, g2, g3, and g4,
# and we wish to constrain g2 & g3 to be equal (note these are in
# positions 2 and 3).

# Generate gg as though from asreml(..., start.values = TRUE)

gg <- data.frame(Vparameter = c("g1", "g2", "g3", "g4"),
                         fac = factor(c(1, 2, 2, 3)))
F <- vcm.lm(~fac, data = gg)
# F
#    fac1 fac2 fac3
# g1    1    0    0
# g2    0    1    0
# g3    0    1    0
# g4    0    0    1
```

---

voltage *Voltage regulators*

Description

Vehicle regulator voltage is measured after setting and testing operations; regulators out of range are returned. A total of 64 regulators were tested at 4 testing stations, and the voltage for individual regulators was set at a total of 10 setting stations. A variable number of regulators (4-8) were set at each station; however, each regulator was tested at every testing station.

Usage

```
voltage
```

Format

A data frame with 4 columns and 256 rows:

- Teststat: Testing station, factor with 4 levels
- Setstat: Setting station voltage treatment, factor with 10 levels
- Regulatr: Number of regulators, factor with 8 levels
- voltage: Reading (in volts)

References

D R Cox, E J Snell (1981). *Applied Statistics; Principles and Examples*. Chapman and Hall, London.

---

vpredict *Functions of variance components*

Description

Calculates means and approximated standard errors from functions of variance components from an `asreml` object.

Usage

```
vpredict(object, xform, ...)
```

Arguments

`object`	An `asreml` object.
`xform`	A two-sided formula, where the left-hand side labels the function name. The right-hand side defines the function to calculate as an algebraic expression.

Details

The standard error of the computed value is approximated using the *Delta* method with the function `deriv()`, which calculates algebraic derivatives for a wide range of expressions. Any of the variance components to be included in the function are represented as: `"V1"`, `"V2"`, `"V3"`, ... in the order they appear in `object$vparameters`. Parentheses, and simple functions like `log`, `exp`, `sqrt` are also allowed. Results are always presented in sigma scale.

Value

A single-row data frame with components:

- `Estimate`: The estimated value of the algebraic expression.
- `SE`: The estimated approximated standard error.

vsn.tryCatch.W.E tryCatch *both warnings (with value) and errors*

Description

Catch and save both errors and warnings, and in the case of a warning, also keep the computed result.

Usage

```
vsn.tryCatch.W.E(expr)
```

Arguments

expr	An R expression to evaluate.

Value

A list with "value" and "warning", where "value" may be an error caught.

Note

This has been altered from the original to return multiple warnings as a list if more than one is thrown in expr and hence has been renamed.

The original was taken from https://stackoverflow.com/questions/32076454/save-object-that-threw-warning-using-trycatch

Original author: Martin Maechler;

Original copyright statement: Copyright (C) 2010-2012 The R Core Team

vsn.tryCatch.W.E.rethrow *Call* tryCatch.W.E *on* expr

Description

Rethrow warnings and errors or return the result of expr.

Usage

```
vsn.tryCatch.W.E.rethrow(expr, call. = NULL)
```

Arguments

expr	An R expression to evaluate.
call.	Replacement text for the $call element of errors and warnings (this can be used to fudge what function has thrown the error/warning, to make things easier for users in some cases). If NULL then the original values are used.

Value

The result of expr.

---

vsr_geom_ribbon *Set mapping on* no_semi_transparency

Description

Simple overload function of ggplot2:geom_ribbon that is used internally in plot functions to either set alpha or colour depending on no_semi_transparency

Usage

```
vsr_geom_ribbon(mapping, no_semi_transparency = FALSE)
```

Arguments

mapping	Passed as the first argument to geom_ribbon.
no_semi_transparency	This should be set to TRUE if the device being used does not support semi-transparency, in which case the ribbon is set with transparency 0 and a grey edge (default = FALSE).

---

vs_active_cores *Return a vector of all the available cores*

Description

Return a vector of all the available cores

Usage

```
vs_active_cores()
```

vs_Call *An overload (of sorts) of the base R function* .Call

Description

An overload (of sorts) of the base R function .Call

Usage

```
vs_Call(
  .NAME,
  ...,
  CORE_VERSION = NULL,
  RETURN_CORE_VERSION = TRUE,
  ERROR_CALL_TEXT = NULL
)
```

Arguments

.NAME	As per .Call (as are all other arguments except CORE_VERSION, RETURN_CORE_VERSION and ERROR_CALL_TEXT).
...	Additional arguments passed to the function being called.
CORE_VERSION	The ID of the core version of asreml that is required. This defines the DLL that is used.
RETURN_CORE_VERSION	If TRUE then an attempt is made to return the core version used as part of the results. This is set to FALSE if the returned object is not a list (default = TRUE).
ERROR_CALL_TEXT	The "call" information reported in warnings and error messages. If NULL then a default of the call from the parent of vs_Call is used (*i.e.*, the call information of the function that called vs_Call).

Value

The results of invoking .Call using the supplied arguments, with .NAME being taken from the DLL associated with CORE_VERSION (see vs_get_core_availability_info() for how the DLL name is obtained).

If the RETURN_CORE_VERSION is TRUE then an additional entry, code_version, is appended to the end of the returned list which holds the ID of the core used.

vs_get_core_availability_info *Information about different* asreml *cores*

Description

Function that returns information relating to the different asreml cores.

Usage

```
vs_get_core_availability_info(core_version = NULL)
```

Arguments

core_version	ID of the core of interest (default = NULL).

Value

If core_version = NULL then returns a vector of all the available core IDs, with the first element being the assumed default value. If core_version is not NULL, then a list is returned, with the name of the shared library (*i.e.* DLL) in dll_name and the version of the core actually used in core_version. (NB: the requested and returned core version will be the same unless the supplied value is invalid, in which case the default core version is assumed.)

vs_get_core_id *Return a core identifier*

Description

Return a core identifier associated with the core version asked for by a user (usually, core_id = core_version, unless core_version is NULL, in which case a default version is assumed and the associated id is returned.

Usage

```
vs_get_core_id(core_version)
```

Arguments

core_version	The version of the core required

vs_get_core_info *Information on the core being used*

Description

Function that returns information on the core being used.

Usage

```
vs_get_core_info(core_version = NULL)
```

Arguments

core_version	ID of the core of interest. If NULL then information for the default core is returned.

wald *Wald test statistics method*

Description

Function that calculates Wald test statistics for fixed effects terms of a fitted model. Presented as a pseudo analysis of variance (ANOVA) using incremental Wald statistics or conditional F-tests. The available method is for asreml class objects.

Usage

```
wald(object, ...)
```

Arguments

object	An object of class asreml.
...	Arguments to wald.asreml

Details

The method wald.asreml() produces two styles of analysis of variance table depending on the settings of denDF and ssType. If denDF = "none" and ssType = "incremental" (the defaults), a pseudo analysis of variance table is returned based on incremental sums of squares with rows corresponding to each of the fixed terms in the object, plus an additional row for the residual. The model sum of squares is partitioned into its fixed term components, and the sum of squares for each term listed in the table of Wald statistics is adjusted for the terms listed in the rows above. The denominator degrees of freedom are not computed and consequently Wald tests are provided.

If either denDF or ssType are not set at their default values, a data frame is returned that will include columns for the approximate denominator degrees of freedom (ddf) and incremental and

conditional approximated F-statistics depending on the combination of options chosen. In all cases, `update.asreml` is called to complete calculations.

The principle used in determining the conditional tests is that a term cannot be adjusted for another term which encompasses it explicitly (for example, **A:C** cannot be adjusted for **A:B:C**) or implicitly (for example, **REGION** cannot be adjusted for **LOCATION** when locations are nested in regions although coded independently).

The numerator degrees of freedom (ndf) for each term is determined as the number of non-singular equations involved in the term. However, the calculation of the ddf is in general not trivial and is computationally expensive. Numerical derivatives require an extra evaluation of the mixed model equations for every variance parameter while algebraic derivatives require a large dense matrix, potentially in the order of the number of equations plus the number of observations. The calculations are suppressed by default.

See Also

`wald.asreml`

`wald.asreml` *Wald test statistics constructor*

Description

Function that calculates Wald test statistics for fixed effects terms of a fitted model. Presented as a pseudo analysis of variance (ANOVA) using incremental Wald statistics or conditional F-tests.

Usage

```
## S3 method for class 'asreml'
wald(object, ..., incr = FALSE)
```

Arguments

`object`	An `asreml` object.
`...`	Arguments to `asreml` can be passed through `update.asreml` if `ssType` is not `"incremental"`.
`incr`	If `TRUE`, only incremental Wald tests are returned irrespective of the `asreml` argument `ssType` (default = `FALSE`).
`denDF`	Controls the suppression ("none") or the use of a particular algorithmic method: `"numeric"` for numerical derivatives, or `"algebraic"` for algebraic derivatives (default = `"none"`).
`ssType`	Controls the tests within the Wald statistics table. These can be tested sequentially (`"incremental"`) or conditional on all the other terms (`"conditional"`) (default = `"conditional"`).

Details

The method `wald.asreml()` produces two styles of analysis of variance table depending on the settings of `denDF` and `ssType`. If `denDF = "none"` and `ssType = "incremental"` (the defaults), a pseudo analysis of variance table is returned based on incremental sums of squares with rows corresponding to each of the fixed terms in the object, plus an additional row for the residual. The model sum of squares is partitioned into its fixed term components, and the sum of squares for each term listed in the table of Wald statistics is adjusted for the terms listed in the rows above. The denominator degrees of freedom are not computed and consequently Wald tests are provided.

If either `denDF` or `ssType` are not set at their default values, a data frame is returned that will include columns for the approximate denominator degrees of freedom (ddf) and incremental and conditional approximated F-statistics depending on the combination of options chosen. In all cases, `update.asreml` is called to complete calculations.

The principle used in determining the conditional tests is that a term cannot be adjusted for another term which encompasses it explicitly (for example, **A:C** cannot be adjusted for **A:B:C**) or implicitly (for example, **REGION** cannot be adjusted for **LOCATION** when locations are nested in regions although coded independently).

The numerator degrees of freedom (ndf) for each term is determined as the number of non-singular equations involved in the term. However, the calculation of the ddf is in general not trivial and is computationally expensive. Numerical derivatives require an extra evaluation of the mixed model equations for every variance parameter while algebraic derivatives require a large dense matrix, potentially in of order of the number of equations plus the number of observations. The calculations are suppressed by default.

Value

A list with class `wald` with the following components:

- `wald`: An `anova` object if `denDF = "none"` and `ssType = "incremental"`, or a data frame otherwise.
- `stratumVariances`: If `denDF` is not `"none"`, a matrix of approximate stratum variances, degrees of freedom and component coefficients is returned for simple variance component models.

References

M G Kenward, J H Roger (1997). "The Precision of Fixed Effects Estimates from Restricted Maximum Likelihood." *Biometrics*, **53**, 983-997.

`wheat` *Wheat variety trial*

Description

Unreplicated early generation wheat variety trial conducted at Tullibigeal in south-western NSW (Australia). The experiment consisted of 525 test lines which were randomly assigned to plots in a 67 row by 10 column array. There was a check plot variety every 6 plots within each column. That is, the check variety was sown on rows 1, 7, 13, . . . , 67 of each column. This variety was numbered 526. A further 6 replicated commercially available varieties (numbered 527 to 532) were also randomly assigned to plots with between 3 to 5 plots for each variety.

Usage

```
wheat
```

Format

A data frame with 5 columns and 670 rows:

- yield: Grain yield (in kg/ha)
- weed: Weed record, a covariate
- Column: Field column position, factor with 10 levels
- Row: Field row position, factor with 67 levels
- Variety: Variety number, factor with 532 levels

Source

NSW Department of Primary Industries

---

where.env *Search for an object*

Description

Function that search for an object in an environment hierarchy.

Usage

```
where.env(name, env = parent.frame(), alt.env = NULL)
```

Arguments

name	The name of the object of interest. If the object is not found then an error message is printed and execution terminated via `stop(...)`.
env	The environment to start searching in (if not found in this environment it traverses up the tree, so it looks next in the parent of env, etc.).
alt.env	An alternative environment to search in, if name is not found in env and `alt.env` is not NULL, then the function will also search the tree starting at `alt.env`.

`wolfinger` *Growth curve study on rats*

Description

Weekly body weights of three treatment groups of rats. A total of 27 rats was divided randomly into 3 groups of 10, 7 and 10, respectively. Group 1 were kept as a control, group 2 had thyroxine and group 3 had thiouracil added to their drinking water. Five weekly measurements were taken on each individual.

Usage

```
wolfinger
```

Format

A data frame with 6 columns and 27 rows:

- Treatment: Water treatment, factor with 3 levels
- wt0: Body weight at week 1
- wt1: Body weight at week 2
- wt2: Body weight at week 3
- wt3: Body weight at week 4
- wt4: Body weight at week 5

References

G E P Box (1950). "Analysis of Growth and Wear Curves." *Biometrics*, **6**, 362-389.

Index