

Package ‘asreml4’

September 16, 2016

Title Fits the Linear Mixed Model

Version 4.0.0.9003

Author David Butler

Maintainer David Butler <dbutler@uow.edu.au>

Description asreml estimates variance components under the linear mixed model using residual maximum likelihood.

Depends R (>= 3.1.0), stats, utils, Matrix

Imports data.table (>= 1.9.6), ggplot2, grid, methods

Suggests lattice, grDevices

License file LICENSE

URL www.vsni.co.uk

LazyData true

ByteCompile yes

RoxygenNote 5.0.1

R topics documented:

ainverse	3
asreml	5
asreml.object	13
asreml.options	15
asreml.read.table	18
asreml4.lic	18
asr_families	19
asr_varioGram	20
asuv	21
binnor	22
captions	23
cheese	23
cheese.cat	24
coef.asreml	24
ebay	25
fitted.asreml	26
grass	26
grassUV	27
harvey	28

harvey.ped	28
harveyg.ped	29
id	30
knownStruc	30
lamb	32
Levels	33
lrt	33
lrt.asreml	34
matern	34
meff	35
meff.asreml	36
metric-1d	36
metric-2d	37
modelFunctions	39
na.method	42
nassau	42
nassau.grm	43
nassau.snp	43
nin89	44
oats	45
orange	45
own	46
plot.asreml	47
plot.varioGram	47
predict.asreml	48
print.asreml.predict	50
print.wald	51
rats	51
residuals.asreml	52
rice	52
riceMV	53
shf	54
sp2mat	55
sp2Matrix	55
str	56
Subset	56
summary.asreml	57
svc	58
svc.asreml	58
timeSeries	59
tr	61
tr.asreml	61
Units	62
unstructured	62
update.asreml	64
varioGram	65
varioGram.asreml	65
vcm.lm	66
voltage	68
vpc.char	68
vpredict	69
vpt.char	69

wald	70
wald.asreml	70
wheat	72
wolfinger	73

Index	74
--------------	-----------

ainverse	<i>Calculate an inverse relationship matrix.</i>
----------	--

Description

Generates an inverse 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 = c("NA", "0", "*"), psort = FALSE)
```

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 use 0 (zero) or NA in the parental columns.
fgen	An optional list of length 2 where <code>fgen[[1]]</code> is a character string naming the column in pedigree that contains the level of selfing or the level of inbreeding of an individual. In <code>pedigree[, fgen[[1]]]</code> , 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 <code>fgen[[2]]</code> , where $0 < \text{fgen}[[2]] < 1$ 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. <code>pedigree[, gender]</code> 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 <i>Fernando and Grossman, 1990</i> 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. The default is $g = 0$.
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 degree of freedom adjustment. The default is $e = 0$.

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 (<i>Dutkowski and Gilmour, 2001</i>). Do not use the selfing argument in conjunction with inBreed or mgs.
inBreed	A numeric scalar (default NA) giving the inbreeding coefficient for <i>base</i> 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.
mgs	If TRUE (default FALSE), the third identity in the pedigree is the male parent of the female parent (maternal grand-sire) rather than the female parent.
mv	A character vector of missing value indicators; elements of pedigree that exactly match any of the members of mv are treated as missing.
psort	If TRUE (default FALSE), the pedigree data frame is returned in founder order after any insertions and permutations.

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 2 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 2 containing the groups and `groupOffset` arguments.

`logdet` The log determinant.

References

Dutkowski GW and Gilmour AR (2001). "Modification of the Additive Relationship Matrix for Open Pollinated Trials." In *Developing the Eucalypt of the Future*, pp. 71. Instituto Forestal Chile, Valdivia.

Fernando R and Grossman M (1990). "Genetic Evaluation with Autosomal and X-Chromosomal Inheritance." *Theoretical and Applied Genetics*, **80**, pp. 75-80.

Meuwissen THE and Luo Z (1992). "Computing Inbreeding Coefficients in Large Populations." *Genetics Selection Evolution*, **24**, pp. 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

pedg <- 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(0.8,0.0,2.0,0.0,2.0,3.0))
pdfg.ai <- ainverse(pdfg,fgen=list('fgen',0.4))
pdfg.mat <- asreml.sparse2mat(pdfg.ai)
zapsmall(solve(pdfg.mat))
zapsmall(cbind(pdfg.a$inbreeding,diag(pdfg.mat)))

## End(Not run)
```

asreml

Fit the linear mixed model.

Description

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

Usage

```
asreml(fixed = y ~ 1, random = ~NULL, sparse = ~NULL, residual = ~NULL,
       G.param = list(), R.param = list(), data = sys.parent(),
       na.action = na.method(), subset, weights, predict = predict.asreml(),
       vcm = vcm.lm(), vcc = matrix(NA), 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(),
       model.frame = TRUE, ...)
```

Arguments

fixed A formula object specifying the fixed terms in the model, with the response on the left of a \sim 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.

random	A formula object specifying the random effects in the model. This argument has the same general characteristics as <code>fixed</code> , but there can be no left side to the <code>~</code> operator. Variance structures imposed on random terms are specified using special model functions.
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 <code>fixed</code> , but there can be no left side to the <code>~</code> expression. Wald statistics are not available for sparse fixed terms in order to reduce the computing load.
residual	A formula object specifying the residual model; any term specified on the left of the <code>~</code> expression is ignored. The default is <code>~units</code> , where the reserved word <code>units</code> is defined as <code>seq(1, nrow(data))</code> and is automatically included in the model frame. Variance models for the residual component of the model can be specified using special model functions. For single-section univariate models, the residual variance model determines the computational mode: If the residual variance model specifies a correlation structure (includes <code>id()</code>), then the model is fitted on the gamma scale, otherwise the model is fitted on the sigma scale. The default is <code>id(units)</code> if not explicitly specified.
G.param	Either, <ul style="list-style-type: none"> • 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 <code>start.values</code> argument; the internal list object is then generated from the contents of this file. <p>On termination, <code>G.param</code> is updated with the final random component estimates.</p>
R.param	Either, <ul style="list-style-type: none"> • 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 <code>start.values</code> argument; the internal list object is then generated from the contents of this file. <p>On termination, <code>R.param</code> is updated with the final residual component estimates.</p>
data	A data frame in which to interpret the variables named in <code>fixed</code> , <code>random</code> , <code>sparse</code> and <code>residual</code> . If the <code>data</code> argument is missing, the default is <code>sys.parent()</code> . The data frame is converted internally to a <code>data.table</code> object and returned as such by <code>model.frame</code> .
na.action	A call to <code>na.method()</code> specifying the action to be taken when missing values are encountered in the response (<code>y</code>) or explanatory variables (<code>x</code>). The function definition for <code>na.method</code> is: <pre>function(y=c("include","omit","fail"), x=c("fail","include","omit"))</pre> <p>The default action is to include (and estimate) missing values in the response, and raise an error if there are missing values in the explanatory variables.</p>
subset	A logical vector identifying which subset of the rows of <code>data</code> should be used in the fit. All observations are included by default.
weights	A character string or name identifying the column of <code>data</code> 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 `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

```

1  1
2  1
2  2
3  1

```

is equivalent to the `vcm` matrix

```

1  0  0
0  1  0
0  2  0
0  0  1

```

- family** A list of functions and expressions for defining the link and variance functions. Optionally a list of such structures for a multivariate analysis involving non-normal variates. Currently this is restricted to a bivariate model where the first variate (excluding the multinomial distribution) is non-normal.
Supported families are *gaussian*, *inverse Gaussian*, *binomial*, *negative binomial*, *poisson*, *Gamma* and *multinomial*. Family objects are generated from the `asreml` [family functions](#) 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.
- 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 2 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 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*. 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 5 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 3: 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.
- If 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 four components: `denDF`, `ssType`, `Ftest` and `kenadj`.
- denDF** a character string from the set `["none", "numeric", "algebraic", "default"]` specifying the calculation of approximate denominator degrees of freedom. The default "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.
- ssType** can be "incremental" for incremental sum of squares (the default) or "conditional" for F-tests that respect both structural and intrinsic marginality.
- Ftest** a formula object 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.
- kenadj** a character string from the set `["none", "expected", "observed"]`, specifying any adjustment to the variance matrix for fixed effects.
- The `wald` argument would typically be set by a call to the `wald` method, which returns more formal output.
- prune** A named list with each component generated from a call to `Subset()`. `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)`. 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()`. `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()`. `uid`, in conjunction with `Units` and the model function `uni()` forms a new factor by selecting a subset of records for 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 not an empty list (the default), `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 in `snp.mat`.

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

<code>model.frame</code>	<p>If TRUE (the default) the model frame (a <code>data.table</code> object with additional attributes derived from the model specification) is included in the returned object. The model frame is required by the <code>asreml</code> <code>summary</code>, <code>plot</code>, <code>resid</code> and <code>fitted</code> methods.</p> <p>In large analyses, the model frame is likely to be a large object. If <code>model.frame</code> is a character string, the model frame is saved in a file as an RDS object by a call to <code>saveRDS()</code>, and named by the supplied string with the extension <code>.RDS</code>. If the model frame is not included in the returned <code>asreml</code> object, this RDS file is searched for by the methods noted above.</p>
<code>...</code>	<p>Additional arguments to <code>asreml</code> from <code>asreml.options()</code>: <code>maxit</code>, <code>workspace</code>, <code>pworkspace</code>, <code>fixgammas</code>, <code>trace</code>, <code>aom</code>.</p>

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. An exception is raised if the response is a factor and `family` is not multinomial.

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 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.

If the response is a matrix, a multivariate linear model is fitted to the columns unless `family = asr_multinomial()` is declared.

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.

`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 model. Instances of generic methods such as `plot()`, `predict()` and `summary()` return various derived results of the fit; `resid()`,

coef() and fitted() extract some of its components. See `asreml.object` for the components of the returned list.

Special 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 categorised as [constructor-type](#) functions, or (default) [identity](#), [time-series](#), [general-structure](#), [1d metric](#), [2d metric](#), [known](#) relationship, general variance structures that span more than one term ([str](#)), or [user-defined](#) structures.

The special model functions that are available in asreml model formulae are introduced below; see the user guide, relevant vignette or the man pages for selected functions for more details or illustration.

The symbols used in the following tables are defined as:

obj	a factor in data.
n	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 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 contr.
lin(obj)	Fits 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 <i>random</i> component of a cubic spline; optionally k knot points.
dev(x)	Fit variate x as a factor; typically used for spline deviations.
ma(obj)	Forms a moving average (1) design matrix from factor obj
at(obj, vec)	Form conditioning covariables for the levels in obj given in vec.
dsum(~term obj, ~model)	Direct sum of term for the levels of obj with variance structure model.
	Used in residual to define multiple <i>sections</i> .
and(obj, k)	Add k times the design matrix for obj to the previous columns.
grp(name)	Include the term defined by name in the group argument in the model.
mbf(name)	Include the covariates defined by name in the mbf argument as a factor.
sbs(name)	Include the term defined in the prune argument in the model.
gpf(name)	Include the term defined in the combine argument in the model.
uni(name)	Include the term defined in the uid argument in the model.

Default identity

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

Time series type 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	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 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)	heterogeneous variance	n		
us(obj)	unstructured variance	$n(n+1)/2$		
sfa(obj, k)	factor analytic; k factors	kn+n		
fa(obj, k)	sparse factor analytic	kn+n		
facv(obj, k)	factor analytic, covariance form	kn+n		
rr(obj, k)	reduced rank variant of fa	kn+n		
chol(obj, k)	Cholesky order k	$(k+1)(n-k)/2$		
cholc(obj, k)	Cholesky	$(k+1)(n-k)/2$		
ante(obj, k)	antependence order k	$(k+1)(n-k)/2$		
mthr(obj)	multinomial models only			

Metric based models in 1D or 2D

Call	Description	p	v	h
exp(x)	exponential 1D	1	2	1+n
iexp(x, y)	isotropic exponential 2D	1	2	1+n
aexp(x, y)	anisotropic exponential 2D	2	3	2+n
gau(x)	gaussian 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
isp(x, y)	isotropic spherical 2D	1	2	1+n
cir(x, y)	isotropic circular 2D	1	2	1+n
mtrn(x, y, ...)	Matern class 2D	*	*	*

*See the user guide for an extended discussion of the Matern class.

Known relationship structures

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

Call	Description
------	-------------

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

User defined structures

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

References

Gilmour AR, Thompson R and Cullis BR (1995). "AI, An Efficient Algorithm for REML Estimation in Linear Mixed Models." *Biometrics*, **51**, pp. 1440-1450.

Kenward MG and Roger JH (1997). "The Precision of Fixed Effects Estimates from Restricted Maximum Likelihood." *Biometrics*, **53**, pp. 983-997.

See Also

[asreml.options](#) [asreml.object](#) [asr_families](#)

Examples

```
data(oats)
oats.asr <- asreml(yield ~ Variety*Nitrogen, random = ~ Blocks/Wplots, data=oats)
```

asreml.object	<i>ASReml object</i>
---------------	----------------------

Description

This (S3) class object contains a fitted linear mixed model from the `asreml()` function. 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 loglikelihood at completion of the `asreml` call.

vparameters The vector of variance parameter estimates from the 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 **Positive**, **Unconstrained** and **Fixed**, respectively. The function `vpc.char` can be used to interpret the numeric values as per `summary.asreml`.

vparameters.type A numeric vector identifying the variance parameter types. Numeric values are used internally and the character codes as used by the `own()` variance model can be obtained from the function `vpt.char`.

vparameters.pc Percentage change in gammas on 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 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 $X(X^T X)^{-1}X^T$ 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`.
- Cfixed** Reflexive generalised inverse of the coefficient matrix of the mixed model equations relating to the dense fixed effects (if `asreml.options()$Cfixed=TRUE`). A matrix of class `Matrix`, sub-class `dspMatrix`.
- Csparse** If `asreml.options()$Csparse` is not `NULL`, the non-zero elements of the reflexive generalised 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 element.
- 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`.
- mef** 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(attributes(object$mf))` for details.

asreml.options	<i>Set asreml options.</i>
----------------	----------------------------

Description

Set less frequently used asreml() options.

Usage

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

Arguments

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

Details

The following settings can be altered:

ai.loadings=-1 Controls modification to AI updates of loadings in extended factor-analytic (fa) 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 (default *i* = -1 specifies 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*.

ai.penalty=4 The algorithm for updating loadings in factor analytic models has been improved. The original update procedure sometimes produced unreasonable updates, or exhibited drift. 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%. The default is $p = 4\%$. After the penalty has reduced to 1%, it is further reduced to 0.2%. ai.penalty can be set to 0 if desired.

ai.scale=0 !AIW in standalone but NOT DOCUMENTED.

ai.sing=FALSE Force continuation if a singularity is detected in the average information matrix.

aodev=FALSE If TRUE, return an analysis of deviance.

aom=FALSE If TRUE, return standardized conditional residuals and standardized conditional BLUPs as additional columns in the coefficients and residuals components.

Cfixed=FALSE If TRUE, return the computed part of the C^{-1} matrix in component Cfixed; the default is FALSE. The inverse coefficient matrix is fully formed for terms in the dense set.

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

debug=FALSE Return internal data structures.

`dense=~NULL` 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.

`design=FALSE` If TRUE, return the design matrix in component `design` of the `asreml` object.

`drop.unused.levels=TRUE` Should unused levels be dropped after a subset type operation?

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

`extra = n` Forces another `mod(n, maxit)` iterations after apparent convergence; the default is `n=1`.

`fail="hard"` If "hard" (the default) fatal errors will terminate execution, otherwise if "soft" such conditions will be reported as warnings, allowing simulation 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.

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

`gammaScale=FALSE` If TRUE (the default is FALSE), 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.

`glmminloop=1` Sets 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 AI 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.

`grid=TRUE` A logical vector of length 1 or `length(design.points)` (see [predict](#)) controlling the expansion of coordinates for 2 dimensional kriging. For a given term, the coordinates for prediction in 2 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 and are taken in parallel.

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

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

`maxit=13` Maximum number of iterations.

`nsppoints=21` 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 $[p, p + 1.1i]$. `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)`.

`pxem=1` (PX)EM update strategy for unstructured (US) variance models when Average Information updates cause them to be non-positive definite (see `uspd`). Valid values are:

pxem	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 steps

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

`pworkspace="128mb"` 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`.

`random.order="noeff"` Reorder terms in the random and sparse formulae in increasing order of 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, or "R" for the default **R** rules.

`scale=1.0` Overall scale parameter

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

`spline.step=list(spl=10000,dev=10000,pol=10000)` 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.

`step.size=0.1` Update shrinkage factor; the step size actually used is `sqrt(step.size)`.

`tol=c(0,0)` 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.

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=TRUE` Report convergence monitoring in the console.

`update.Gcon=FALSE` Update the constraint status of variance parameters derived from the random formula on termination.

`update.Rcon=FALSE` Update the constraint status of variance parameters derived from the residual formula on termination.

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

`workspace="128mb"` 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).

asreml.read.table	<i>Read in a data frame.</i>
-------------------	------------------------------

Description

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](#).

asreml4.lic	<i>Check the asreml license.</i>
-------------	----------------------------------

Description

Checks the status of the asreml license and reports any errors.

Usage

```
asreml4.lic(license = "asreml.lic", quiet = FALSE)
```

Arguments

license	A character string naming the license file; the default is "asreml.lic". Currently ignored.
quiet	If FALSE (the default) a string containing the license details is echoed to the console.

Details

asreml4.lic() checks for a valid license in the default location for the current platform (Windows, Linux or Mac) and raises a **tltk** dialogue box if a license file is not found, or the license is invalid. A license file is created and stored locally if the user has access to a valid network FLEXlm license server, or has an activation code from vsni.co.uk.

Value

A character string containing the license owner, serial number, expiry date and time remaining is silently returned.

asr_families *GLM family objects for asreml.*

Description

Family functions specify the details of the models accepted by the family argument to asreml.

Usage

```
asr_gaussian(link = "identity", dispersion = NA)
asr_Gamma(link = "inverse", dispersion = 1, phi = 1)
asr_inverse.gaussian(link = "1/mu^2", dispersion = NA)
asr_binomial(link = "logit", dispersion = 1, total = NULL)
asr_multinomial(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; valid values are: Gaussian: identity, log, inverse Gamma: identity, log, inverse inverse.gaussian: 1/mu^2, identity, log, inverse binomial: logit, probit, cloglog multinomial: logit, probit, cloglog negative.binomial: identity, log, inverse poisson: identity, log, sqrt
dispersion	If NA, the default for Gaussian and inverse Gaussian models, the dispersion parameter is estimated, otherwise it is fixed at the nominated value (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 (default).
- asr_Gamma: The gamma model.
- asr_inverse.gaussian: The inverse Gaussian model.

- `asr_binomial`: The binomial model. 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` is `NULL`, a binary (0,1) response is expected.
- `asr_multinomial`: The multinomial model. The response can either be a matrix of counts with the response categories as columns, with an additional column for the total number of cases in each row, or in univariate style with the response as a factor. If the response is a matrix and `total=NULL`, the total counts are calculated from the category columns.
- `asr_negative.binomial`: The negative-binomial model.
- `asr_poisson`: The poisson model.

<code>asr_varioGram</code>	<i>Empirical variogram.</i>
----------------------------	-----------------------------

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

<code>x</code>	Numeric vector of x coordinates, may also be a matrix or data frame with 2 or 3 columns. If <code>ncol(x)</code> is 3, the columns are taken to be the x and y coordinates and the response (z), respectively. If <code>ncol(x)</code> is 2, the columns are taken to be the x coordinates and the response, respectively. In this case the y coordinates are generated as <code>rep(1, nrow(x))</code> .
<code>y</code>	Numeric vector of y coordinates.
<code>z</code>	The response vector.
<code>composite</code>	For data on a regular grid. If <code>TRUE</code> , 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.
<code>model</code>	Can only be "empirical" at present.
<code>metric</code>	The distance between (x,y) points. Valid measures are "euclidean" or "manhattan".
<code>angle</code>	A vector of directions. Angles are measured in degrees anticlockwise from the x axis. Default is 0.
<code>angle.tol</code>	The angle subtended by each direction. That is, an arc angle \pm <code>angle.tol/2</code> . Default is 180 which gives an omnidirectional variogram.
<code>nlag</code>	The maximum number of lags; default is 20.
<code>maxdist</code>	The fraction of the maximum distance to include in the calculation. The default is half the maximum distance in the data.
<code>xlag</code>	The width of the lags. If missing, <code>xlag</code> is set to <code>maxdist/nlag</code> .
<code>lag.tol</code>	The distance tolerance. If missing, <code>lag.tol</code> is set to <code>xlag/2</code> .
<code>grid</code>	If <code>FALSE</code> , forces polar variograms if (x,y) specifies a regular grid; default is <code>TRUE</code> .

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 is assumed sorted with the x coordinates changing the fastest; the data is 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

Webster W and Oliver MA (2001). *Geostatistics for Environmental Scientists*. John Wiley: West Sussex.

asuv	<i>Univariate data frame.</i>
------	-------------------------------

Description

Make a univariate dataframe 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 univariate data frame object.
- response** A character string to be used as the name of the concatenated response vector; the default is "y".
- traitName** A character string to name the resulting column of multi-variate response names; the default is "trait".
- traitsWithinUnits** The sort order of the returned data frame. The default, TRUE, orders the data as multi-variate response traits nested within experimental units.

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.

 binnor

Footrot scores on lambs.

Description

Incidence of two footshape classes on 2513 lambs.

Usage

binnor

Format

A data frame with 10 columns and 2513 rows:

record An integer vector of record numbers

year Cross year, an integer vector with values 1 and 2

Grp A factor with 5 levels

Sex A factor with 2 levels

Sire A factor with 18 levels

score4 Incidence of footshape class 1

score5 Incidence of footshape class 2

scald Incidence of scald disease

rot Incidence of rot disease

norm A random normal variate

Details

The feet of 2513 lambs born in 1980 and 1981 from 5 mating groups were scored in two footshape 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.

References

Gilmour AR, Anderson RD and Rae AL (1985). "The Analysis of Binomial Data by a Generalized Linear Mixed Model." *Biometrika*, **72**, pp. 593-599.

captions	<i>Axis captions</i>
----------	----------------------

Description

Vector of axis labels for each residual type.

Usage

captions

Format

A named vector of axis label strings.

cheese	<i>Cheese tasting.</i>
--------	------------------------

Description

Tasting category counts on four cheeses.

Usage

cheese

Format

A data frame with 11 columns and 4 rows:

Cheese A 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

Details

Four cheeses were scored on a nine point scale by 52 judges.

References

McCullagh P and Nelder JA (1989). *Generalized Linear Models*. CRC Press.

cheese.cat	<i>Cheese tasting incidence.</i>
------------	----------------------------------

Description

Tasting incidences on four cheeses.

Usage

```
cheese.cat
```

Format

A data frame with 2 columns and 208 rows:

Taste A factor with 9 levels

Cheese A factor with 4 levels

Details

Four cheeses were scored on a nine point scale by 52 judges. This is the [cheese](#) data in *incidence* form, where each taste category and cheese combination is recorded.

References

McCullagh P and Nelder JA (1989). *Generalized Linear Models*. CRC Press.

coef.asreml	<i>Extract model coefficients</i>
-------------	-----------------------------------

Description

Extract model coefficients from an asreml object.

Usage

```
## S3 method for class 'asreml'
coef(object, list = FALSE, pattern = character(0))
```

Arguments

object	An asreml object.
list	If TRUE, the coefficients are returned in a named list of length the number of terms in the model; default FALSE.
pattern	A term in the model as a character string; if an interaction then separate the term names in the string by “:”. A regular expression is constructed from pattern to extract a subset of coefficients.

Value

If neither `pattern` nor `list` is set, then a list of length 3 with the following components:

fixed solutions to the mixed model equations for the fixed (dense) terms.

random E-BLUPs for the effects in the random model.

sparse solutions to the mixed model equations for the fixed sparse-stored terms.

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; otherwise a single column matrix of effects as specified by `pattern`.

Examples

```
data(oats)
oats.asr <- asreml(yield ~ Variety*Nitrogen, random = ~ Blocks/Wplots, data=oats)
coef(oats.asr)
coef(oats.asr, list=TRUE)
coef(oats.asr, pattern="Blocks:Wplots")
```

 ebay

Nested correlated observations.

Description

Hypothetical bids within 15 auctions.

Usage

```
ebay
```

Format

A data frame with 3 columns and 2500 rows:

seq An integer vector of record numbers

auc An integer vector of auction identifiers

val Numeric vector of bid values

fitted.asreml	<i>Extract fitted values.</i>
---------------	-------------------------------

Description

Extracts fitted values from an asreml object.

Usage

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

Arguments

object	An asreml object.
type	if "link", the linear fit on the link scale, otherwise, if "response", the fitted values obtained by transforming the linear predictors by the inverse link function.

Value

A numeric vector of fitted values.

Examples

```
data(oats)
oats.asr <- asreml(yield ~ Variety*Nitrogen, random = ~ Blocks/Wplots, data=oats)
fitted(oats.asr)
```

grass	<i>Plant height</i>
-------	---------------------

Description

Plant height measurements on 14 plants at 5 occasions.

Usage

```
grass
```

Format

A data frame with 7 columns and 14 rows:

Tmt A factor with 2 levels, diseased or healthy

Plant A 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

Details

The 14 plants were either diseased or healthy and were arranged in a glasshouse in a completely random 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.

Source

J. Lamptey, Rothamsted Experimental Station, UK.

grassUV	<i>Plant height (univariate form)</i>
---------	---------------------------------------

Description

Plant height measurements on 14 plants at 5 occasions.

Usage

grassUV

Format

A data frame with 5 columns and 70 rows:

Tmt A factor with 2 levels, diseased or healthy

Plant A factor with 14 levels

Time A factor with 5 levels

HeightID A factor with 5 levels

y Plant height

Details

The 14 plants were either diseased or healthy and were arranged in a glasshouse in a completely random 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.

Source

J. Lamptey, Rothamsted Experimental Station, UK.

 harvey

Weights of cattle.

Description

Average daily gain for 65 Hereford steers.

Usage

harvey

Format

A data frame with 8 columns and 65 rows:

Calf A factor with 65 levels

Sire A factor with 9 levels

Dam A factor with 1 level

Line A factor with 3 levels

ageOfDam An integer vector

y1 Age at weaning

y2 Initial weight

y3 Average daily gain ($\times 100$)

Details

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.

References

Harvey WR (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 cattle.

Description

Crossing history for 65 Hereford steers.

Usage

harvey.ped

Format

A data frame with 3 columns and 65 rows:

Calf An integer vector

Sire A character vector

Dam An integer vector

Details

Calf, sire and dam identities for 65 Hereford steers; the first three columns of [harvey](#).

References

Harvey WR (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 cattle.

Description

Crossing history for 65 Hereford steers with genetic groups.

Usage

harveyg.ped

Format

A data frame with 3 columns and 77 rows:

Calf A character vector

Sire A character vector

Dam A character vector

Details

Calf, sire and dam identities for 65 Hereford steers in 3 genetic groups. Additional rows to [harvey.ped](#) describe the genetic group structure.

References

Harvey WR (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](#)

id	<i>Default identity variance models.</i>
----	--

Description

Model functions for identity variance models.

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", "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`).

Functions

- `asr_idv`: Identity variance model.
- `asr_idh`: Heterogeneous identity (diagonal) variance model.

knownStruc	<i>Known variance structures.</i>
------------	-----------------------------------

Description

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

Usage

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

```
ide(obj)
```

Arguments

obj	A factor in data.
source	The known inverse or relationship matrix: <ul style="list-style-type: none"> • a matrix inheriting class <code>ginv</code> such as that produced from a call to <code>ainverse</code>. This is a known inverse variance matrix held in three column co-ordinate form in row major order. <code>source</code> must have a <code>rowNames</code> attribute. • a matrix (or <code>Matrix</code> object) with a <code>dimnames</code> 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 attribute <code>INVERSE</code> set to <code>TRUE</code>. • a numeric vector of the lower triangular elements in row major order. The vector must have a <code>rowNames</code> attribute, and if an inverse structure, it must also have an <code>INVERSE</code> attribute set to <code>TRUE</code>.
singG	Ignored if <code>source</code> has class <code>ginv</code> or attribute <code>INVERSE=TRUE</code> ; in such cases <code>source</code> must be one of: <ul style="list-style-type: none"> • a sparse matrix in coordinate form with class <code>ginv</code>, or attribute <code>INVERSE=TRUE</code>, or • an object of class <code>matrix</code> or <code>Matrix</code> with <code>INVERSE=TRUE</code>), or • a vector assumed to be the lower triangle in row major order with attribute <code>INVERSE=TRUE</code>. <p>If <code>source</code> does not have class <code>ginv</code>, or the attribute <code>INVERSE</code> is <code>FALSE</code> or is not set, and <code>singG</code> is <code>NULL</code> (the default), then <code>source</code> is assumed a positive definite relationship matrix and <code>singG</code> is reset to <code>"PD"</code>. Otherwise, a character string giving the state of the (to be inverted) source object:</p> <p><code>"PD"</code> positive definite (default)</p> <p><code>"ND"</code> source is negative definite, but not singular. <code>asreml</code> ignores the indefinite condition and proceeds</p> <p><code>"PSD"</code> source is positive semi-definite. In this case, <code>asreml</code> proceeds using lagrangian multipliers to process the matrix. Two cases arise: whether the singularity arises because of an effect has zero variance or whether it arises as a linear dependence. An example of the first is when the GRM 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</p> <p><code>"NSD"</code> source is negative semi-definite. Negative roots are ignored, and the zero roots are handled as for <code>"PSD"</code>.</p>

Details

If `source` inherits from class `Matrix`, `asreml` will convert `source` internally to either 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_olddide`: 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.

 lamb

Footrot counts on lambs.

Description

Counts of two footshape classes on 2513 lambs.

Usage

lamb

Format

A data frame with 12 columns and 68 rows:

cyr Cross year, an integer vector with values 1 and 2

Grp A factor with 5 levels

Sex A factor with 2 levels

Sire A factor with 18 levels

xxx A numeric vector

tot Binomial totals for each sex, sire, group combination

15 Incidence of footshape class 2

14 Incidence of footshape class 1

ls Incidence of scald disease

lr Incidence of rot disease

prop Footshape class 2 (15) as a proportion

fail Footshape class 2 *failure* counts (1-15)

Details

The feet of 2513 lambs born in 1980 and 1981 from 5 mating groups were scored in two footshape 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 is grouped into 68 sex, sire and mating group combinations.

References

Gilmour AR, Anderson RD and Rae AL (1985). "The Analysis of Binomial Data by a Generalized Linear Mixed Model." *Biometrika*, **72**, pp. 593-599.

Levels	<i>Combine factor levels.</i>
--------	-------------------------------

Description

Forms a new model term from an existing factor by merging a subset of its levels.

Usage

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

Arguments

f	A factor in the data.
x	A vector of length <code>length(levels(f))</code> defining the levels of f to merge. See the combine argument to asreml .

lrt	<i>Likelihood ratio tests.</i>
-----	--------------------------------

Description

Generic function to calculate likelihood ratio statistics for fitted models. The available method is for `asreml` class objects.

Usage

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

Arguments

...	A sequence of <code>asreml</code> objects as comma separated names.
-----	---

See Also

[lrt.asreml](#)

lrt.asreml	<i>REML Likelihood ratio tests</i>
------------	------------------------------------

Description

Extracts the REML log likelihood and numbers of variance parameters from a sequence of asreml objects, and computes a sequence of pairwise likelihood ratio tests.

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.
boundary	If TRUE (the default) hypothesized parameter values being tested lie on the boundary of the parameter space.

Details

The models are arranged in increasing order of number of variance parameters and assumed nested in this sequence. If the reduced model is obtained by setting positively-constrained variance parameters in the full model to zero, set boundary to 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 likelihood ratio statistic, degrees of freedom and number of parameters.

References

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

matern	<i>Matern variance structure.</i>
--------	-----------------------------------

Description

Model function for an extended Matern class.

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: 2 is Euclidean distance (default), and 1 is city block.
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

The `mtrn` special function implements an extended Matern class which accomodates geometric anisotropy and a choice of metrics for random fields observed in two dimensions (*Haskard et al., 2007*). See the User Guide for 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_mtrnv`: Matern variance model, homogeneous variance form.
- `asr_mtrnh`: Matern variance model, heterogeneous variance form.

References

Haskard KA, Cullis BR and Verbyla AP (2007). “Anisotropic Matern correlation and spatial prediction using REML.” *Journal of Agricultural and Biological Sciences*, **12**, pp. 147-160.

meff

Marker effects.

Description

Generic function to compute genetic marker effects for a fitted model and matrix of marker scores. The available method is for `asreml` class objects.

Usage

```
meff(x, ...)
```

Arguments

x	An object of class <code>asreml</code> .
...	Arguments to <code>mef.asreml</code>

See Also

[meff.asreml](#)

meff.asreml	<i>Marker effects.</i>
-------------	------------------------

Description

Calculate regressor (marker) effects using the original regressor scores and an asreml fit using the known subject 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 default, the term associated with the first variance matrix in the mef list will be used.
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, calculate the standard errors of the effects; default FALSE.
evaluate	If TRUE (the default), evaluate the effects by a call to update.asreml , otherwise return the unevaluated call.
...	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.

metric-1d	<i>Metric based models in one dimension.</i>
-----------	--

Description

Metric based variance model functions in one dimension.

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)
```

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).

Details

Includes one dimensional exponential and gaussian power models (exp, gau).

Functions

- asr_expv: Homogeneous variance form.
- asr_exph: Heterogeneous variance form.
- asr_gau: Gaussian power model.
- asr_gauv: Gaussian power model, homogeneous variance form.
- asr_gauh: Gaussian power model, heterogeneous variance form.

 metric-2d

Metric based models in two dimensions.

Description

Metric based variance model functions in two dimensions.

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)
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

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

Details

Includes two two dimensional isotropic exponential, gaussian, euclidean, spherical and circular power models (`iexp`, `igau`, `ieuc`, `sph`, `cir`), anisotropic exponential and gaussian models (`aexp`, `agau`) and the Matern class (`mtrn`).

Functions

- `asr_iexpv`: Homogeneous variance form.

- `asr_iexph`: Heterogeneous variance form.
- `asr_aexp`: Anisotropic exponential variance model.
- `asr_aexpv`: Anisotropic exponential variance model, homogeneous variance form.
- `asr_aexph`: Anisotropic exponential variance model, heterogeneous variance form.
- `asr_igau`: Isotropic Gaussian variance model.
- `asr_igauv`: Isotropic Gaussian variance model, homogeneous variance form.
- `asr_igauh`: Isotropic Gaussian variance model, heterogeneous variance form.
- `asr_agau`: Anisotropic Gaussian variance model.
- `asr_agauv`: Anisotropic Gaussian variance model, homogeneous variance form.
- `asr_agauh`: Anisotropic Gaussian variance model, heterogeneous variance form.
- `asr_ieuc`: Isotropic Euclidean variance model.
- `asr_ieucv`: Isotropic Euclidean variance model, homogeneous variance form.
- `asr_ieuch`: Isotropic Euclidean variance model, heterogeneous variance form.
- `asr_sph`: Spherical variance model.
- `asr_sphv`: Spherical variance model, homogeneous variance form.
- `asr_sphh`: Spherical variance model, heterogeneous variance form.
- `asr_cir`: Circular variance model.
- `asr_cirv`: Circular variance model, homogeneous variance form.
- `asr_cirh`: Circular variance model, heterogeneous variance form.

 modelFunctions

Model term constructor functions.

Description

This class of special functions constructs model terms with specific properties.

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=0, 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 <code>asreml()</code> <code>mbf</code> list argument.
grp	A component name from the <code>asreml()</code> <code>group</code> list argument.
p	The exponent in a power function term (<code>pow</code>).
offset	Constant added to <code>obj</code> ; default 0.
t	pol: The maximum degree of a set of orthogonal polynomials formed from <code>obj</code> . If negative, the intercept polynomial is omitted. leg: The maximum degree of a set of Legendre polynomials formed from <code>obj</code> . If negative, the intercept polynomial is omitted.
k	The number of equally spaced knot points for a cubic smoothing spline. If zero or omitted, <code>k</code> is set to <code>asreml.options()\$knots</code> (default 50).
init	Optional initial value for the default identity variance model (<code>idv</code>) when used in the random formula.
lvls	Vector of levels of the conditioning factor (<code>obj</code>) that define the conditioning covariates formed by <code>at</code> . If numeric, <code>lvls</code> indexes the levels vector of <code>obj</code> ; that is, <code>levels(obj)[lvls]</code> .
times	Multiples (may be non-integer) of the design matrix for <code>obj</code> are added to the preceding design matrix.
model	A formula of the form $\sim A+B \dots Z$, where <code>A</code> and <code>B</code> define variance matrices for simple or compound model terms, and <code>Z</code> is a simple conditioning factor whose levels identify and determine the number of sub-matrices in the direct sum. The <code>" "</code> operator is applied associatively and operates with all terms on its left; that is, <code>A+B C</code> implies <code>(A+B) C</code> and is equivalent to <code>A C+B C</code> .
levels	A list of length the number of terms in the left hand side of <code>model</code> that are separated by <code>"+"</code> . The components of <code>levels</code> are vectors of factor levels of <code>Z</code> . If there is only one term in the left hand side of <code>model</code> (or if the context allows, see examples) then <code>levels</code> may be a vector. If <code>NULL</code> , the default is to use <code>levels(Z)</code> .
outer	if <code>TRUE</code> , independent blocks of correlated observations are modelled with common variance and correlation parameters; the blocks can be of different sizes.
contr	An integer vector of contrast coefficients parallel to <code>levels(obj)</code> .

Functions

- `asr_con`: Sum to zero constraints.
- `asr_lin`: Create a variate from `obj`.
- `asr_pow`: Creates 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 order 1 design matrix from `obj`.
- `asr_at`: Form a conditioning covariable 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 columns of data.
- `asr_dsum`: Direct sum structures for residual models.
- `asr_C`: Treatment contrasts

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
residual = ~ dsum(~ ar1(Column):ar1(Row) + id(Column):ar1(Row) | Site,
  levels = list(c(1,3), c(2,4)))

## equivalent
residual = ~ dsum(~ ar1(Column):ar1(Row) | Site, levels=c(1,3))
  + dsum(~ id(Column):ar1(Row) | Site, levels=c(2,4))

## "biological" Date within Plot
residual = ~ dsum(~ ar1(Date) | Plot, outer=TRUE)

## "explicit" times
residual = ~ dsum(~ exp(Date) | Plot, outer=TRUE)

## End(Not run)
```

na.method	<i>Missing value action for asreml dataframes.</i>
-----------	--

Description

Function to deal with missing values in the data argument to asreml. "include" retains NAs in the data, "omit" drops records with NAs and "fail" raises 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 is "include".
x	Action to take if missing values are present in covariates; default is "fail".

Value

A list with components x and y.

nassau	<i>Clone data.</i>
--------	--------------------

Description

Height of Loblolly pine trees at age 6.

Usage

```
nassau
```

Format

A data frame with 5 columns and 6795 rows:

Rep Factor with 8 levels
IncBlock Factor with 80 levels
CultureID Factor with 2 levels
clonefv Factor with 860 levels
ht6 Tree height

References

Resende MF, Munoz P, Resende MD, Garrick DJ, Fernando RL, Davis JM, Jokela EJ, Martin TA, Peter GF and Kirst M (2012). "Accuracy of Genomic Selection Methods in a Standard Data Set of Loblolly Pine (*Pinus taeda* L.)." *Genetics*, **190**, pp. 1503-1510.

nassau.grm	<i>Genetic relationship matrix.</i>
------------	-------------------------------------

Description

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

Usage

```
nassau.grm
```

Format

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

See Also

[nassau nassau.snp](#)

nassau.snp	<i>Genetic marker scores.</i>
------------	-------------------------------

Description

Genetic marker matrix for 963 Loblolly pine clones and 4854 snp markers.

Usage

```
nassau.snp
```

Format

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

See Also

[nassau nassau.grm](#)

nin89

Advanced breeding trial.

Description

An advanced Nebraska Intrastate Nursery (NIN) breeding trial conducted at Alliance in 1988/89.

Usage

nin89

Format

A data frame with 11 columns and 242 rows:

Variety A factor with 56 levels

Id A factor with 56 integral levels

pid A numeric variate containing plot numbers

raw Plot weights

Rep A factor with 4 levels

nloc Trial location (4)

yield Grain yield in t/ha

lat Spatial coordinate

long Spatial coordinate

Row A factor with 22 levels

Column A factor with 11 levels

Details

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 randomised complete block (RCB) design.

References

Stroup WW, Baenziger PS and Mulitze DK (1994). "Removing Spatial Variation from Wheat Yield Trials: A Comparison of Methods." *Crop Science*, **86**, pp. 62-66.

oats

Split plot design.

Description

Yield of oats with four fertilizer treatments.

Usage

oats

Format

A data frame with 6 columns and 72 rows:

Blocks Complete field replicates, factor with 6 levels

Nitrogen Factor with 4 levels

Subplots Factor with 4 levels

Variety Factor with 3 levels

Wplots Factor with 3 levels

yield Grain yield in grams

Details

The yield of oats from a split-plot field trial with three varieties and four 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).

References

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

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 A factor with 5 levels

x Age of tree

circ Trunk circumference (mm)

Season A factor with 2 levels

References

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

 own

User-defined variance models.

Description

Specify an external function that provides a variance matrix, or its inverse, and the respective derivative matrices with respect to the parameters.

Usage

```
own(obj, fun = "myowngdg", init = NA, type = character(0))
```

Arguments

obj A factor in data.

fun The name (as a character string) of an R function to compute the variance matrix and its derivatives. This function must accept two input arguments:

order A scalar giving the dimension of the structure being defined;

param A numeric vector of $1, \dots, k$ parameter values;

and return a list of $k+1$ matrices: the variance matrix, or its inverse, followed by the k derivative matrices. This list may have an attribute `INVERSE`, a logical scalar identifying the structures as variance matrices or their inverses; if `INVERSE` is absent the default is `FALSE`.

init The k -vector of parameter values.

type A character vector defining the type of each parameter from the set "V", "G", "R", "C", "P" and "L", identifying each parameter as type *variance*, *variance-ratio*, *correlation*, *covariance*, *positive correlation* or *loading*, respectively.

Details

The own variance model allows users to specify external variance structure(s). This requires the user to provide an R function that accepts the current set of parameters, forms the variance matrix and a full set of derivative matrices, and return these in a list object. The R function may invoke compiled code if necessary. Before each iteration, `asreml` calls the nominated function with the current parameter estimates to update the variance matrix and derivatives.

plot.asreml	<i>Plot diagnostics for an asreml object.</i>
-------------	---

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(object, res = "default", spatial = "trend",
      facet = FALSE)
```

Arguments

object	An asreml object.
res	The type of residuals; see residuals.asreml .
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" (the default) then units are not added even if present in the model.
facet	If TRUE, multi-panel conditioning plots are produced for models with multi-section residual structures.

Details

If the residual structure of the model contains multiple sections, the default plots are conditioned on the factor whose levels define the sections. For multivariate analyses, the plots are conditioned on trait.

Value

An invisible list of ggplot2 objects.

plot.varioGram	<i>Plot a variogram.</i>
----------------	--------------------------

Description

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

Usage

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

Arguments

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

Value

An invisible lattice object.

<code>predict.asreml</code>	<i>Predict linear functions of effects.</i>
-----------------------------	---

Description

An instance of the generic method `predict` for objects of class `asreml`. 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 = list(), 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 <code>asreml.knot.points</code> argument and included in the <code>predict</code> set with the <code>predict.design.points</code> argument. The factors <code>mv</code> and <code>units</code> 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 <code>classify</code> set but not those in the <code>average</code> 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.

<code>ignore</code>	A character vector specifying which variables to ignore in forming the predictions.
<code>use</code>	A character vector specifying which variables to add to the prediction model after the default rules have been invoked.
<code>except</code>	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 <code>except</code> list.
<code>only</code>	A character vector specifying which variables (only) form the prediction model, that is, the default rules are not invoked.
<code>associate</code>	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 <i>left-to-right</i> outer to inner nesting order. Nested hierarchies are separated by the "+" operator; only one "+" operator is currently permitted, giving a maximum of two <code>associate</code> lists.
<code>average</code>	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.
<code>vcov</code>	If TRUE (default FALSE), the full variance-covariance matrix of the predicted values is returned in a component <code>vcov</code> .
<code>sed</code>	If TRUE (default FALSE), the full standard error of difference matrix of the predicted values is returned in a component <code>sed</code> .
<code>parallel</code>	If TRUE (default FALSE), the levels of the <code>classify</code> factors given in the <code>levels</code> list are expanded in parallel; in this case levels must be specified for all factors in the <code>classify</code> set, and they must be of equal length.
<code>aliased</code>	If TRUE (default FALSE), the predicted values are returned for non-estimable functions.
<code>design.points</code>	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 <code>spl()</code> , <code>pol()</code> , <code>dev()</code> or metric type models. If an element of this list is a list of length 2 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 <code>design.points</code> must match exactly those used in the model functions.
<code>evaluate</code>	If FALSE (the default is TRUE), an unevaluated call to <code>update.asreml</code> is returned, otherwise the call is evaluated. Setting <code>evaluate = FALSE</code> returns a list that may be used with the <code>predict</code> argument in a call to <code>asreml</code> .
<code>...</code>	Additional arguments to <code>asreml</code> .

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.

vcov Optional variance-covariance matrix of class `dspMatrix` of the predicted values.

avsed Summary standard error of difference.

References

Welham SJ, Cullis BR, Gogel BJ, Gilmour AR and Thompson R (2004). "Prediction in linear mixed models." *Australian and New Zealand Journal of Statistics*, **46**, pp. 325-347.

`print.asreml.predict` *Print predictions.*

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.

print.wald	<i>Print a wald object.</i>
------------	-----------------------------

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

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

rats	<i>Reproductive study on rats</i>
------	-----------------------------------

Description

Litter weights of pups from rats given three doses (control, low and high) of an experimental compound affecting reproductive performance.

Usage

```
rats
```

Format

A data frame with 6 columns and 322 rows:

Dose A factor with 3 levels

Sex A factor with 2 levels

littersize A covariate

Dam A factor with 27 levels

Pup A factor with 18 levels

weight Individual pup weights in grams

Details

Thirty female rats (dams) were randomly split into three groups of 10 and each group randomly assigned a dosing level. Three litters had to be dropped from the high dose level.

References

Dempster AP, Patel CM, Selwyn MR and Roth AJ (1984). "Statistical and Computational Aspects of Mixed Model Analysis." *Journal of the Royal Statist. Series C.*, **33**(2), pp. 203-214.

residuals.asreml	<i>Extract model residuals.</i>
------------------	---------------------------------

Description

Extracts residuals from asreml objects.

Usage

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

Arguments

object	An asreml object.
type	Type of residuals: "deviance", "pearson", "working", "response", "response" or "stdCond". Default is "deviance".
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 spatial = "plot"; the default is spatial = "trend".

Value

A numeric vector containing the model residuals.

rice	<i>Rice bloodworms.</i>
------	-------------------------

Description

An investigation of the tolerance of rice varieties to attack by the larvae of bloodworms.

Usage

```
rice
```

Format

A data frame with 6 columns and 264 rows:

Pair A factor with 132 levels

rootwt Root weight

Run A factor with 66 levels

sqrtroot The square root of rootwt

Tmt A factor with 2 levels

Variety A factor with 44 levels

Details

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 three replicates of each. The variety concurrence was such that: eight varieties occurred with only one other variety, 22 with two other varieties and the remaining 14 with three different varieties.

References

Stevens MM, Fox KM, Warren GN, Cullis BR, Coombes NE and Lewin LG (1999). "An Image Analysis Technique for Assessing Resistance in Rice Cultivars to Root-feeding Chironomid Midge Larvae (Diptera: Chironomidae)." *Field Crops Research*, **66**, pp. 25-26.

riceMV

Rice bloodworms (multivariate form).

Description

An investigation of the tolerance of rice varieties to attack by the larvae of bloodworms.

Usage

riceMV

Format

A data frame with 7 columns and 132 rows:

Pair A factor with 132 levels

Run A factor with 66 levels

Variety A 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

Details

Multivariate form of [rice](#) with root weight in two variates corresponding to the levels of the applied treatment (control or exposed to bloodworms).

References

Stevens MM, Fox KM, Warren GN, Cullis BR, Coombes NE and Lewin LG (1999). "An Image Analysis Technique for Assessing Resistance in Rice Cultivars to Root-feeding Chironomid Midge Larvae (Diptera: Chironomidae)." *Field Crops Research*, **66**, pp. 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.

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

Details

The trial was a balanced lattice with 25 varieties in 6 replicates, arranged in a 15 column by 10 row grid of plots.

References

Kempton RA and Fox PN (1997). *Statistical Methods for Plant Variety Evaluation*. Chapman and Hall.

sp2mat	<i>Convert sparse matrix.</i>
--------	-------------------------------

Description

Convert a sparse matrix in three column coordinate form to a dense matrix.

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	<i>Convert sparse matrix.</i>
-----------	-------------------------------

Description

Convert 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 default is FALSE), the result is stored as a dense symmetric <code>dspMatrix</code> object, otherwise a sparse symmetric matrix.
triplet	If TRUE (the default is FALSE), the result is a <code>CsparseMatrix</code> , otherwise a <code>TsparseMatrix</code> (triplet form).

Details

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

str	<i>General variance structures.</i>
-----	-------------------------------------

Description

General variance structure spanning consecutive model terms.

Usage

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

Arguments

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

Details

Typically, a variance structure applies to an individual term (main effect or interaction) in the linear model, and there is no covariance between model terms. Sometimes it is appropriate to include a covariance, such as random coefficients regression, for example. 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`.

Subset	<i>Subset a factor.</i>
--------	-------------------------

Description

Forms a new model term from an existing factor by selecting a subset of its levels.

Usage

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

Arguments

f	A factor in the data.
x	A character or numeric vector of levels to select. See the <code>prune</code> argument to asreml .

summary.asreml	<i>Summarize an asreml object</i>
----------------	-----------------------------------

Description

A summary method for objects inheriting from class asreml.

Usage

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

Arguments

object	An asreml object.
scale	if "sigma" (the default), random parameter values are reported on the <i>sigma</i> scale only, otherwise, if "gamma", an additional column of variance ratios is returned.
coef	If TRUE (default is FALSE), the coefficients and their standard errors are included in the return object.
vparameters	If TRUE (default is FALSE), the variance parameters are included in the return object in list form.

Value

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

call The call component from object.

loglik The loglik component from object.

nedf The nedf component from object.

sigma $\sqrt{\text{object}\$sigma2}$.

varcomp A dataframe summarising the random parameter vector (object\$vparameters). Variance component ratios are included if scale="gamma", and a measure of precision is included along with boundary constraints at termination and the percentage change in the final iteration.

aic Akaike information criterion.

bic Bayesian information criterion.

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 (deviance/nedf) if object\$deviance != 0.

coef.fixed A matrix of coefficients and their standard errors for fixed effects if coef=TRUE.

coef.random A matrix of coefficients and their standard errors for random effects if coef=TRUE.

coef.sparse A matrix of coefficients and their standard errors for sparse-fixed effects 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.

svc	<i>Stratum variance coefficients method.</i>
-----	--

Description

Generic function to calculate stratum variance coefficients for a variance component model. The available method is for asreml class objects.

Usage

```
svc(x, ...)
```

Arguments

x	An object of class asreml.
...	Arguments to svc.asreml

See Also

[svc.asreml](#)

svc.asreml	<i>Stratum variance coefficients</i>
------------	--------------------------------------

Description

A method to compute the approximate stratum variances for simple variance component models.

Usage

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

Arguments

object	An object of class asreml.
order	The sequence in which to consider the random terms: may be one of "as.is" or "noeff" for the order as given in the asreml object, or ordered by increasing number of effects, respectively. The default is "as.is". Note that the default ordering for random terms in asreml is "noeff".

Details

Approximate stratum variances and degrees of freedom for simple variance components models are returned. For the 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*, asreml computes approximate stratum degrees of freedom and stratum variances by a modified Cholesky diagonalisation of the average information matrix.

References

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

timeSeries

Time series type variance models.

Description

Time series type correlation and variance models.

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

<code>obj</code>	A factor in data.
<code>init</code>	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.

Details

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 3 (`sar2`), moving average models of order 1 and 2 (`ma1`, `ma2`) and the autoregressive-moving average model (`arma`).

Functions

- `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.
- `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.
- `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.
- `asr_sarv`: Symmetric autoregressive model; homogeneous variance form.
- `asr_sarh`: Symmetric autoregressive model; heterogeneous variance form.
- `asr_sar2`: Constrained autoregressive model of order 3.
- `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.
- `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.
- `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.
- `asr_armav`: Autoregressive-moving average model; homogeneous variance form.
- `asr_armah`: Autoregressive-moving average model; heterogeneous variance form.

tr	<i>Trace convergence.</i>
----	---------------------------

Description

Generic function to graph convergence history for a fitted model. The available method is for `asreml` class objects.

Usage

```
tr(x, ...)
```

Arguments

x	An object of class <code>asreml</code> .
...	Arguments to <code>tr.asreml</code>

See Also

[tr.asreml](#)

tr.asreml	<i>Convergence trace for asreml objects.</i>
-----------	--

Description

Scatter plots of component value against iteration for each random component.

Usage

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

Arguments

object	An <code>asreml</code> 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.

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	<i>Unit level numbers.</i>
-------	----------------------------

Description

Forms a new model term from an existing factor by choosing a subset of its record numbers.

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. See the uid argument to asreml .

unstructured	<i>General structure variance models.</i>
--------------	---

Description

General correlation and covariance models.

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)
```

```
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

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

Details

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`).

Functions

- `asr_corv`: Simple correlation model, homogeneous variance form.
- `asr_corh`: Simple correlation model, heterogeneous variance form.
- `asr_corb`: Banded correlation model with `b` bands.
- `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.
- `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_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_rrr`: Factor analytic model with `k` factors; reduced rank formulation of `fa()` where the default boundary constraints for the specific variances are set to **Fixed**.

update.asreml	<i>Update an asreml model.</i>
---------------	--------------------------------

Description

Extract and evaluate the call from the fitted object, replacing any arguments with changed values. In particular, `G.param` and `R.param` are automatically updated with those stored in the object.

Usage

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

Arguments

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

Either a new updated `asreml` object, else an unevaluated expression for creating such an object.

Examples

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

## End(Not run)
```

varioGram	<i>Empirical variogram method.</i>
-----------	------------------------------------

Description

Generic function to calculate an empirical variogram. The available method is for asreml class objects.

Usage

```
varioGram(x, ...)
```

Arguments

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

See Also

[varioGram.asreml](#)

varioGram.asreml	<i>Empirical variogram constructor.</i>
------------------	---

Description

Calculate 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 <code>asreml</code> .
type	Type of residuals, see <code>residuals.asreml</code> .
spatial	Whether to include a nugget effect; see <code>residuals.asreml</code> .
formula	An optional model formula designed to extract <i>residuals</i> from the <i>random</i> component of the model rather than the <i>residual</i> component. This is a two sided formula where the response is a pattern in the style required by the <code>pattern</code> argument of <code>coef.asreml</code> .
composite	The argument to <code>asr_varioGram</code> .
model	The argument to <code>asr_varioGram</code> .
metric	The argument to <code>asr_varioGram</code> .
angle	The argument to <code>asr_varioGram</code> .
angle.tol	The argument to <code>asr_varioGram</code> .
nlag	The argument to <code>asr_varioGram</code> .
maxdist	The argument to <code>asr_varioGram</code> .
xlag	The argument to <code>asr_varioGram</code> .
lag.tol	The argument to <code>asr_varioGram</code> .
grid	The argument to <code>asr_varioGram</code> .

Details

Calls `asr_varioGram` to calculate the empirical semi-variogram.

Examples

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

## End(Not run)
```

vcm.lm

Specify constraints among variance parameters.

Description

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

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 (the default), unused levels in factors are removed.
intercept	if FALSE (the default), the intercept is not included in the call to model.matrix when forming the constraints matrix.
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 M from a simple linear model. Let κ be the n_k vector of unconstrained variance parameters and T be a $n_k \times n_c$ matrix imposing the linear constraints $T^T \kappa = s$. This is equivalent to $\kappa = M\theta + Es$ where θ is the n_c vector of constrained parameters.

The matrix M is given as the value to the vcm argument of asreml. M 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 n_k names of the variance parameters is returned by asreml when start.values=TRUE. The matrix M is obtained from a call to model.matrix using form and additional factors derived from or interacting with Vparameter.

Value

A $n_k \times n_c$ matrix M specifying the variance parameter constraints, where n_c is the length of the reduced vector of variance parameters. In the simplest case, for example, where two parameters are constrained to be equal, $n_c = n_k - 1$ and the appropriate column of M will contain ones in the rows corresponding to the parameters in question and zeros elsewhere.

References

Smith AB (1999). *Multiplicative Mixed Models*. PhD thesis, Department of Statistics, University of Adelaide.

Examples

```
# Suppose there are 4 variance parameters: g1, g2, g3, and g4,
# and we wish to constrain 2 & 3 to be equal

# 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)))

M <- vcm.lm(~fac, data=gg)
# M
#   fac1 fac2 fac3
# g1   1   0   0
# g2   0   1   0
# g3   0   1   0
# g4   0   0   1
```

voltage	<i>Voltage regulators</i>
---------	---------------------------

Description

Vehicle regulator voltage is measured after setting and testing operations; regulators out of range are returned.

Usage

voltage

Format

A data frame with 4 columns and 256 rows:

Teststat A factor with 4 levels

Setstat A factor with 10 levels

Regulatr A factor with 8 levels

voltage Reading in volts

Details

Sixty-four regulators were tested at four 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.

References

Cox DR and Snell EJ (1981). *Applied Statistics; Principles and Examples*. Chapman and Hall, London.

vpc.char	<i>Variance parameter constraint codes</i>
----------	--

Description

Character vector of variance parameter constraint codes corresponding to the numeric values returned in the `vparameters.con` component of the `asreml` object.

Usage

`vpc.char(object)`

Arguments

object An `asreml` object with a `vparameters.con` component; the vector of numeric variance parameter constraint codes.

Value

A character vector of constraint codes. Common values are "P" (=1), "U" (=3) and "F" (=4) for positive, unrestricted and fixed, respectively.

vpredict	<i>Functions of variance parameters.</i>
----------	--

Description

Form functions of variance components from an `asreml` object.

Usage

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

Arguments

<code>object</code>	An <code>asreml</code> object.
<code>xform</code>	A two-sided formula, where the left-hand side labels the estimate. The right-hand side defines the derived parameter as an algebraic expression. Any of the estimated parameters can be included, represented as "V1", "V2", "V3", ... in the order they appear in <code>object\$vparameters</code> . Parentheses, and simple functions like <code>log</code> , <code>exp</code> , <code>sqrt</code> are allowed.

Details

The standard error of the computed value is calculated using the *delta* method using `deriv()`, which calculates algebraic derivatives for a wide range of expressions. The variance components are represented in the expression by "V1", "V2", "V3", etc.

Value

A single-row data frame with components:

Estimate The result of the algebraic expression.

SE The estimated standard error.

vpt.char	<i>Variance parameter type codes</i>
----------	--------------------------------------

Description

Character vector of variance parameter type codes corresponding to the numeric values returned in the `vparameters.type` component of the `asreml` object.

Usage

```
vpt.char(object)
```

Arguments

`object` An `asreml` object with a `vparameters.type` component; the vector of numeric variance parameter type codes.

Value

A character vector of type codes from the set "V", "G", "R", "C", "P" and "L", identifying each parameter as type *variance*, *variance ratio*, *correlation*, *covariance*, *positive correlation* or *loading*, respectively.

wald	<i>Wald statistics method.</i>
------	--------------------------------

Description

Generic function to calculate Wald statistics for a fitted model. The available method is for `asreml` class objects.

Usage

```
wald(x, ...)
```

Arguments

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

See Also

[wald.asreml](#)

wald.asreml	<i>Wald test constructor for asreml objects.</i>
-------------	--

Description

Pseudo analysis of variance using incremental Wald statistics or conditional F-tests.

Usage

```
## S3 method for class 'asreml'
wald(object, Ftest = formula("~NULL"), denDF = c("none",
  "default", "numeric", "algebraic"), ssType = c("incremental",
  "conditional"), kenadj = c("none", "expected", "observed"), ...)
```

Arguments

object	An asreml object.
Ftest	A one sided formula of the form <code>~ test-term background-terms</code> specifying a conditional Wald test of the contribution of <code>test-term</code> conditional on those listed in <code>background-terms</code> , and the those in the random and sparse model formulae.
denDF	Compute approximate denominator degrees of freedom: can be "none" (the default) to suppress the computations, "numeric" for numerical methods, "algebraic" for algebraic methods or "default" to automatically choose numeric or algebraic computations depending on problem size. The denominator degrees of freedom are calculated according to <i>Kenward and Roger (1997)</i> for fixed terms in the dense part of the model.
ssType	Can be "incremental" for incremental sum of squares (the default) or "conditional" for F-tests that respect both structural and intrinsic marginality.
kenadj	Can be "none" to compute Wald statistics using an unadjusted variance matrix for the fixed effects, "expected" to adjust for <i>expected</i> information, or "observed" to adjust for <i>observed</i> information.
...	Arguments to <code>asreml</code> can be passed through <code>update.asreml</code> if <code>ssType</code> is not "incremental".

Details

`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 and incremental and conditional F statistics depending on the combination of options chosen. `update.asreml` is called to complete the 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). See the vignette for further information.

The numerator degrees of freedom for each term is determined as the number of non-singular equations involved in the term. However, the calculation of the denominator df 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 of order 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

Kenward MG and Roger JH (1997). "The Precision of Fixed Effects Estimates from Restricted Maximum Likelihood." *Biometrics*, **53**, pp. 983-997.

wheat

Wheat variety trial

Description

Unreplicated early generation wheat variety trial conducted at Tullibigeal in south-western NSW.

Usage

wheat

Format

A data frame with 5 columns and 670 rows:

yield Grain yield in kg/ha

weed A covariate

Column A factor with 10 levels

Row A factor with 67 levels

Variety A factor with 532 levels

Details

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 of each.

Source

NSW Department of Primary Industries

wolfinger

Growth curve study on rats

Description

Weekly body weights of three treatment groups of rats.

Usage

wolfinger

Format

A data frame with 6 columns and 27 rows:

Treatment A 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

Details

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 thyroxin and group 3 had thiouracil added to their drinking water. Five weekly measurements were taken on each individual.

References

Box GEP (1950). "Analysis of Growth and Wear Curves." *Biometrics*, **6**, pp. 362-389.

Index

*Topic **datasets**

- binnor, 22
- captions, 23
- cheese, 23
- cheese.cat, 24
- ebay, 25
- grass, 26
- grassUV, 27
- harvey, 28
- harvey.ped, 28
- harveyg.ped, 29
- lamb, 32
- nassau, 42
- nassau.grm, 43
- nassau.snp, 43
- nin89, 44
- oats, 45
- orange, 45
- rats, 51
- rice, 52
- riceMV, 53
- shf, 54
- voltage, 68
- wheat, 72
- wolfinger, 73

1d metric, 11

2d metric, 11

aexp (metric-2d), 37

aexph (metric-2d), 37

aexpv (metric-2d), 37

agau (metric-2d), 37

agauh (metric-2d), 37

agauv (metric-2d), 37

ainverse, 3

and (modelFunctions), 39

ante (unstructured), 62

ar1 (timeSeries), 59

ar1h (timeSeries), 59

ar1v (timeSeries), 59

ar2 (timeSeries), 59

ar2h (timeSeries), 59

ar2v (timeSeries), 59

ar3 (timeSeries), 59

ar3h (timeSeries), 59

ar3v (timeSeries), 59

arma (timeSeries), 59

armah (timeSeries), 59

armav (timeSeries), 59

asr_aexp (metric-2d), 37

asr_aexph (metric-2d), 37

asr_aexpv (metric-2d), 37

asr_agau (metric-2d), 37

asr_agauh (metric-2d), 37

asr_agauv (metric-2d), 37

asr_and (modelFunctions), 39

asr_ante (unstructured), 62

asr_ar1 (timeSeries), 59

asr_ar1h (timeSeries), 59

asr_ar1v (timeSeries), 59

asr_ar2 (timeSeries), 59

asr_ar2h (timeSeries), 59

asr_ar2v (timeSeries), 59

asr_ar3 (timeSeries), 59

asr_ar3h (timeSeries), 59

asr_ar3v (timeSeries), 59

asr_arma (timeSeries), 59

asr_armah (timeSeries), 59

asr_armav (timeSeries), 59

asr_at (modelFunctions), 39

asr_binomial (asr_families), 19

asr_C (modelFunctions), 39

asr_chol (unstructured), 62

asr_cir (metric-2d), 37

asr_cirh (metric-2d), 37

asr_cirv (metric-2d), 37

asr_con (modelFunctions), 39

asr_cor (unstructured), 62

asr_corb (unstructured), 62

asr_corbh (unstructured), 62

asr_corbv (unstructured), 62

asr_corg (unstructured), 62

asr_corgh (unstructured), 62

asr_corgv (unstructured), 62

asr_corh (unstructured), 62

asr_corv (unstructured), 62

asr_dev (modelFunctions), 39

- asr_diag (unstructured), 62
 - asr_dsum (modelFunctions), 39
 - asr_exp (metric-1d), 36
 - asr_exph (metric-1d), 36
 - asr_expv (metric-1d), 36
 - asr_fa (unstructured), 62
 - asr_facv (unstructured), 62
 - asr_families, 13, 19
 - asr_Gamma (asr_families), 19
 - asr_gau (metric-1d), 36
 - asr_gauh (metric-1d), 36
 - asr_gaussian (asr_families), 19
 - asr_gauv (metric-1d), 36
 - asr_grp (modelFunctions), 39
 - asr_id (id), 30
 - asr_idh (id), 30
 - asr_idv (id), 30
 - asr_ieuc (metric-2d), 37
 - asr_ieuch (metric-2d), 37
 - asr_ieucv (metric-2d), 37
 - asr_iexp (metric-2d), 37
 - asr_iexph (metric-2d), 37
 - asr_iexpv (metric-2d), 37
 - asr_igau (metric-2d), 37
 - asr_igauh (metric-2d), 37
 - asr_igauv (metric-2d), 37
 - asr_inverse.gaussian (asr_families), 19
 - asr_leg (modelFunctions), 39
 - asr_lin (modelFunctions), 39
 - asr_ma (modelFunctions), 39
 - asr_ma1 (timeSeries), 59
 - asr_ma1h (timeSeries), 59
 - asr_ma1v (timeSeries), 59
 - asr_ma2 (timeSeries), 59
 - asr_ma2h (timeSeries), 59
 - asr_ma2v (timeSeries), 59
 - asr_mbf (modelFunctions), 39
 - asr_mtrn (matern), 34
 - asr_mtrnh (matern), 34
 - asr_mtrnv (matern), 34
 - asr_multinomial (asr_families), 19
 - asr_negative.binomial (asr_families), 19
 - asr_olddide (knownStruc), 30
 - asr_own (own), 46
 - asr_poisson (asr_families), 19
 - asr_pol (modelFunctions), 39
 - asr_pow (modelFunctions), 39
 - asr_rr (unstructured), 62
 - asr_sar (timeSeries), 59
 - asr_sar2 (timeSeries), 59
 - asr_sar2h (timeSeries), 59
 - asr_sar2v (timeSeries), 59
 - asr_sarh (timeSeries), 59
 - asr_sarv (timeSeries), 59
 - asr_sfa (unstructured), 62
 - asr_sph (metric-2d), 37
 - asr_sphh (metric-2d), 37
 - asr_sphv (metric-2d), 37
 - asr_spl (modelFunctions), 39
 - asr_str (str), 56
 - asr_us (unstructured), 62
 - asr_varioGram, 20, 66
 - asr_vm (knownStruc), 30
 - asreml, 5, 33, 56, 62
 - asreml.object, 11, 13, 13
 - asreml.options, 10, 13, 15
 - asreml.read.table, 18
 - asreml4.lic, 18
 - asuv, 21
 - at (modelFunctions), 39
- binnor, 22
- C (modelFunctions), 39
 - captions, 23
 - cheese, 23, 24
 - cheese.cat, 24
 - chol (unstructured), 62
 - cir (metric-2d), 37
 - cirh (metric-2d), 37
 - cirv (metric-2d), 37
 - coef (coef.asreml), 24
 - coef.asreml, 24, 66
 - con (modelFunctions), 39
 - constructor-type, 11
 - cor (unstructured), 62
 - corb (unstructured), 62
 - corbh (unstructured), 62
 - corbv (unstructured), 62
 - corg (unstructured), 62
 - corgh (unstructured), 62
 - corgv (unstructured), 62
 - corh (unstructured), 62
 - corv (unstructured), 62
- dev (modelFunctions), 39
 - diag (unstructured), 62
 - dsum (modelFunctions), 39
- ebay, 25
 - exp (metric-1d), 36
 - exph (metric-1d), 36
 - expv (metric-1d), 36
- fa (unstructured), 62

- facv (unstructured), 62
- family functions, 7
- fitted.asreml, 26

- gau (metric-1d), 36
- gauh (metric-1d), 36
- gauv (metric-1d), 36
- general-structure, 11
- grass, 26
- grassUV, 27
- grp (modelFunctions), 39

- harvey, 28, 29
- harvey.ped, 28, 29
- harveyg.ped, 29

- id, 30
- ide (knownStruc), 30
- identity, 11
- idh (id), 30
- idv (id), 30
- ieuc (metric-2d), 37
- ieuch (metric-2d), 37
- ieucv (metric-2d), 37
- iexp (metric-2d), 37
- iexpv (metric-2d), 37
- igau (metric-2d), 37
- igauh (metric-2d), 37
- igauv (metric-2d), 37

- known, 11
- knownStruc, 30

- lamb, 32
- leg (modelFunctions), 39
- Levels, 33
- lin (modelFunctions), 39
- lrt, 33
- lrt.asreml, 33, 34

- ma (modelFunctions), 39
- ma1 (timeSeries), 59
- ma1h (timeSeries), 59
- ma1v (timeSeries), 59
- ma2 (timeSeries), 59
- ma2h (timeSeries), 59
- ma2v (timeSeries), 59
- matern, 34
- mbf (modelFunctions), 39
- meff, 35
- meff.asreml, 36, 36
- metric-1d, 36
- metric-2d, 37

- modelFunctions, 39
- mtrn, 38
- mtrn (matern), 34
- mtrnh (matern), 34
- mtrnv (matern), 34

- na.method, 42
- nassau, 42, 43
- nassau.grm, 43, 43
- nassau.snp, 43, 43
- nin89, 44

- oats, 45
- orange, 45
- own, 46

- plot (plot.asreml), 47
- plot.asreml, 47
- plot.varioGram, 47
- pol (modelFunctions), 39
- pow (modelFunctions), 39
- predict, 16
- predict.asreml, 48, 50
- print.asreml.predict, 50
- print.wald, 51

- rats, 51
- read.table, 18
- resid (residuals.asreml), 52
- residuals.asreml, 47, 52, 66
- rice, 52, 53, 54
- riceMV, 53
- rr (unstructured), 62

- sar (timeSeries), 59
- sar2 (timeSeries), 59
- sar2h (timeSeries), 59
- sar2v (timeSeries), 59
- sarh (timeSeries), 59
- sarv (timeSeries), 59
- sfa (unstructured), 62
- shf, 54
- sp2mat, 55
- sp2Matrix, 55
- sph (metric-2d), 37
- sphh (metric-2d), 37
- sphv (metric-2d), 37
- spl (modelFunctions), 39
- str, 11, 56
- Subset, 56
- summary, 14
- summary (summary.asreml), 57
- summary.asreml, 57

svc, 58
svc.asreml, 58, 58

time-series, 11
timeSeries, 59
tr, 61
tr.asreml, 61, 61

Units, 62
unstructured, 62
update (update.asreml), 64
update.asreml, 36, 49, 64
us (unstructured), 62
user-defined, 11

varioGram, 65
varioGram.asreml, 65, 65
vcm.lm, 66
vm (knownStruc), 30
voltage, 68
vpc.char, 68
vpredict, 69
vpt.char, 69

wald, 70
wald.asreml, 51, 70, 70
wheat, 72
wolfinger, 73