Navigating from ASReml-R Version 3 to 4

User Transition Guide

D G Butler[1], B J Gogel[2], B R Cullis[1] and R Thompson[3]

[1]National Institute for Applied Statistics and Research, Australia
School of Mathematics and Applied Statistics
University of Wollongong

[2]School of Agriculture, Food and Wine, University of Adelaide, Australia

[3]Centre for Mathematical and Computational Biology, and Department of Biomathematics and Bioinformatics, Rothamsted Research, Harpenden AL5 2JQ, United Kingdom

September 12, 2018

Navigating from ASReml-R Version 3 to 4

D G Butler, B J Gogel, B R Cullis and R Thompson

Published by

VSN International Ltd,
2 Amberside House,
Wood Lane,
Hemel Hempstead,
HP2 4TP UK.
email: info@asreml.co.uk
website: http://www.vsni.co.uk/

Copyright Notice

The correct bibliographical reference for this document is:

Butler, D. G., Gogel, B.G., Cullis, B.R. and Thompson, R. 2017. Navigating from ASReml-R Version 3 to 4. VSN International Ltd, Hemel Hempstead, HP1 1ES, UK.

Author email addresses

dbutler@uow.edu.au
beverley.gogel@adelaide.edu.au
bcullis@uow.edu.au
robin.thompson@rothamsted.ac.uk

Preface

Developments in ASReml-R Version 4

There are a number of new developments in Version 4. They include a more sensible and consistent nomenclature, a more unified framework for model specification and output objects, and extended functionality. Specific enhancements include:

- Computationally efficient fitting of random regression models when there are more variables than observations: motivated by the use of SNP marker data to explain genotypes.
- Fitting linear relationships among variance structure parameters.
- Computing functions of variance components and their approximate standard errors.
- Calculating information criteria.
- Easier, more consistent and more useful specification of direct sum structures for residual models. These occur when data observations are partitioned into sections to which separate variance structures are applied. For example, separate spatial structures and residual error variances would typically be specified for each site in a multi-environment trial (MET) analysis.
- Simpler, more consistent specification and fitting of known variance matrices, including relationship matrices, and allowance for singular matrices.
- Introduction of the own() variance model to allow the specification of a user-defined variance structure.
- Extensions to generalized linear models including threshold models and bivariate models with one variate having a normal distribution and the other variate distributed from an exponential family distribution.
- Generating design matrices to allow use of derived model terms and functions.
- Introduction of functions to generate factors that either combine levels of a factor or use a subset of levels to allow easier prediction of models.

A list of terms in Version 3 with their replacement terms in Version 4 is presented in Chapter 2. Changes to the `asreml()` function with Version 4 are described in Chapter 3 which commences with a brief overview of the theory underpinning linear mixed models followed by separate sections covering arguments, the `asreml` object, screen output, methods and functions, and lists. The features that are new with Version 4 are described in Chapter 4.

Who should read this document?

This document is for existing users who wish to keep abreast of the developments with Version 4. Users who are new to ASReml-R with this version are referred to the ASReml-R Reference Manual Version 4. A suite of both introductory and more advanced vignettes will also be available at `http://asreml.org`.

Companion documents

Two main documents currently partner this navigation document: the ASReml-R Reference Manual Version 4 which is an update of the Version 3 reference manual (Butler et al.; 2009) and the ASReml-R package reference which is a pdf version of the ASReml-R help pages obtained in R by typing `help(asreml)`. Both documents are available at `http://asreml.org` under `Resources` > `ASReml docs` and on the VSN International website `https://www.vsni.co.uk`.

Contents

List of Tables

1 Getting started

1.1 ASReml-R Version 4.1

The package is named `asreml` and can be installed along side Version 3 (V3), although the root function remains `asreml()` so that both versions cannot be loaded into an R session concurrently.

1.2 Installation and loading

Please refer to the VSN International website https://www.vsni.co.uk for information on installation and loading.

1.3 The `asreml` license

Please refer to the VSN International website https://www.vsni.co.uk for information on the ASReml-R license.

2 What's been replaced in ASReml-R Version 4

As ASReml-R has developed it has become apparent that some terms can be named more appropriately, the naming of others can be simplified and in some cases the terms can be grouped together. Table 2.1 is a list of terms in Version 3 that have been removed, deprecated or deleted, with their replacement and action in Version 4. Reasons for the change(s) are also given.

Table 2.1: List of terms (arguments, functions, objects, methods) in Version 3 with their status, action, replacement term and reason for replacement in Version 4

terms in Release 3	status in Release 4	action in Release 4	replacement in Release 4	reasons for change/notes
`asreml()` arguments				
`as.multivariate`	removed	terminates call	`asmv`	more succinct argument name
`constraints`	removed	terminates call	`vcc`, `vcm`, 4.1	extended functionality
`control`	removed	terminates call	use `asreml.options`, 3.6	sensible grouping of job control and less frequently used options
`dump.model`	removed	terminates call		obsolete
`gammas.table`	removed	terminates call	use `vparameters.table`	more appropriate name
`ginverse`	removed	terminates call	use `vm()`, 4.2	more unified framework for specification of special variance models
`maxiter`	deprecated	honoured	`maxit`	can be set in `asreml.options`
`model`	removed	terminates call		obsolete
`na.method.X`	removed	terminates call	use `na.action`, 3.2	more unified framework for related arguments
`na.method.Y`	removed	terminates call	use `na.action`, 3.2	
`rcov`	removed	terminates call	`residual`, 3.1.1	more appropriate name
`asreml.`*argument*	removed	terminates call	`asr_`*argument*, 3.2	prefix `asreml.` replaced by `asr_` to simplify name and avoid confusion, eg. GLM family functions, *argument* = `gaussian`, `Gamma`, `inverse.gaussian`, `binomial`, `multinomial`, `negative.binomial`, `poisson`
`predictpoints`	deleted	terminates call	`design.points`, 3.2	more appropriate name
`pwrpoints`	deleted	terminates call	`pwr.points`, 3.2	more appropriate name
`splinepoints`	deleted	terminates call	`knot.points`, 3.2	more appropriate name
`splinescale`	deleted	terminates call	`spline.scale`, 3.6	more appropriate name

2 What's been replaced in ASReml-R Version 4

List of terms (arguments, functions, objects, methods) in Version 3, their status and actions in Version 4 with their replacement terms and reasons for replacement in Version 4

terms in Release 3	status in Release 4	action in Release 4	replacement in Release 4	reasons for change/notes
Mixed model functions				
`at()` in `rcov` or `residual`	removed	terminates call	use `dsum`, 4.2	avoid inconsistent use of `at()`
`giv`	removed	R error will be generated	use `vm()`, 4.2	more unified framework for specification of special variance models
`ped`	removed	R error will be generated	use `vm()`, 4.2	more unified framework for specification of special variance models
The `asreml` object				
`family`	removed	reports `NULL`	`family`	sensible inclusion in the model frame
`gammas`	removed	reports `NULL`	`vparameters`	more appropriate name
`gammas.type`	removed	reports `NULL`	`vparameters.type`	more appropriate name
`gammas.con`	removed	reports `NULL`	`vparameters.con`	more appropriate name
`monitor`	removed	reports `NULL`	`trace`	logical companion name to the `tr()` method
`fixed.formula`	removed	reports `NULL`	use `formulae`, 3.7	a list with components `fixed`, `random`, `sparse` and `residual`; more unified framework for related components of the `asreml` object
`random.formula`	removed	reports `NULL`	use `formulae`, 3.7	more unified framework for related components of the `asreml` object
`sparse.formula`	removed	reports `NULL`	use `formulae`, 3.7	more unified framework for related components of the `asreml` object
Methods or functions				
`asreml.Ainverse()`	removed	R error will be generated	`ainverse()`, 3.6	more appropriate function name
`asreml.control()`	removed	R error will be generated	`asreml.options`, 3.6	now a function that groups job control and less frequently used options
`asreml.variogram()`	removed	R error will be generated	use `asr_varioGram`, 3.6	renamed to avoid conflict with similar naming in other R packages
`variogram()`	removed	R error will be generated	`varioGram()`, 3.6	renamed to avoid conflict with similar naming in other R packages
`variogram.asreml()`	removed	R error will be generated	`varioGram.asreml()`, 3.6	renamed to avoid conflict with similar naming in other R packages

3 What's changed in ASReml-R Version 4

In this chapter we describe changes to the `asreml()` function with Version 4. To provide the theoretical base and facilitate an understanding of some of the terminology used in describing changed/new features, we commence with an algebraic outline of the linear mixed models framework.

3.1 Specification of the linear mixed model

If $\boldsymbol{y}$ denotes the vector of observations, the general linear mixed model fitted by ASReml-R can be written as

$$\boldsymbol{y} = \boldsymbol{X}\boldsymbol{\tau} + \boldsymbol{Z}\boldsymbol{u} + \boldsymbol{e}$$

where $\boldsymbol{\tau}$ is a vector of fixed effects, $\boldsymbol{X}$ is the design matrix that associates observations with the appropriate combination of fixed effects, $\boldsymbol{u}$ is a vector of random effects, $\boldsymbol{Z}$ is the design matrix that associates observations with the appropriate combination of random effects and $\boldsymbol{e}$ is the vector of residual errors.

ASReml-R assumes the vectors $\boldsymbol{u}$ and $\boldsymbol{e}$ are uncorrelated with each other and have variance matrices $\text{var}(\boldsymbol{u}) = \boldsymbol{G}(\boldsymbol{\sigma}_g)$ and $\text{var}(\boldsymbol{e}) = \boldsymbol{R}_v(\boldsymbol{\sigma}_r)$ that are functions of variance parameters $\boldsymbol{\sigma}_g$ and $\boldsymbol{\sigma}_r$. The variance matrix for $\boldsymbol{y}$ is then of the form

$$\text{var}(\boldsymbol{y}) \quad = \quad \boldsymbol{Z}\boldsymbol{G}(\boldsymbol{\sigma}_g)\boldsymbol{Z}^{\mathsf{T}} + \boldsymbol{R}_v(\boldsymbol{\sigma}_r)$$

which we will refer to as the *sigma parameterization* of the G and R variance structures, and the individual variance structure parameters in $\boldsymbol{\sigma}_g$ and $\boldsymbol{\sigma}_r$ will be referred to as *sigmas*. Other parameterizations are possible and are sometimes useful. Motivated by mixed models when $\boldsymbol{R}_v(\boldsymbol{\sigma}_r)$ can be written as a scaled correlation matrix $\boldsymbol{R}_v(\sigma_r) = \boldsymbol{\sigma}_e^2\boldsymbol{R}_c(\boldsymbol{\gamma}_r)$, we can then write $\boldsymbol{G}(\boldsymbol{\sigma}_g) = \sigma_e^2\boldsymbol{G}(\boldsymbol{\gamma}_g)$ and $\text{var}(y) = \sigma_e^2(\boldsymbol{Z}\boldsymbol{G}(\boldsymbol{\gamma}_g)\boldsymbol{Z}^{\mathsf{T}} + \boldsymbol{R}_c(\boldsymbol{\gamma}_r))$. We call this the *gamma parameterization* and the individual variance structure parameters in $\boldsymbol{\gamma}_g$ and $\boldsymbol{\gamma}_r$ will be referred to as *gammas*. Variance parameters in $\boldsymbol{\sigma}_g$, for example, σ_{gi} are re-parameterized as variance ratios $\gamma_{gi} = \sigma_{gi}/\sigma_e^2$. Correlation parameters are the next most common type of variance parameter in $\boldsymbol{G}$ and $\boldsymbol{R}_c$ and then $\gamma_{gi} = \sigma_{gi}$ and $\gamma_{ri} = \sigma_{ri}$. It is usually easier to suggest initial values of the parameters for the gamma parameterization, that is, for the gammas, and no initial value for the scaling parameter $\boldsymbol{\sigma}_e^2$ is needed.

3.1.1 `rcov` becomes `residual`

Perhaps one of the most notable changes (for existing users) from Version 3 to 4 is the change of the argument name for specifying the variance structure for the residual error term from `rcov` to `residual`. The default variance structure for data comprising a single section is still `id(units)`. However, the special function `dsum` is now used to specify a direct sum structure for data that is partitioned into sections, for example, in multi-environment trial (MET) analysis, see Section 4.2.

3.1.2 Switching between the gamma and sigma parameterization

For single section models when the residual model can be expressed as a scaled residual matrix, ASReml-R offers the option of fitting parameters using either the sigma parameterization (with sigma parameters, see Section 3.1) or the gamma parameterization (with gamma parameters). Specifying the residual model as a variance structure (or with `dsum` for multi-section models, see end of section) forces ASReml-R to use the sigma parameterization. For example:

```
residual = ~idv(units)
residual = ~ar1v(Column):ar1(Row)
residual = ~us(Trait):units
```

would all use the sigma parameterization for model fitting. The following is the likelihood convergence and table of variance parameter estimates for the RCB example when an IDV variance structure is specified for the residual model:

```
rcb.asr <- asreml(fixed = yield~1 + Variety, random = ~ idv(Block),
residual = ~idv(units), data = rcb.dat)

Model fitted using the sigma parameterization.
ASReml 4.1.0 Fri Feb  2 09:36:56 2018
          LogLik        Sigma2     DF     wall    cpu
 1      -28.7733           1.0     96 09:36:56    0.0
 2        2.7153           1.0     96 09:36:56    0.0
 3       35.8161           1.0     96 09:36:56    0.0
 4       55.9947           1.0     96 09:36:56    0.0
 5       64.0512           1.0     96 09:36:56    0.0
 6       65.0829           1.0     96 09:36:56    0.0
 7       65.1126           1.0     96 09:36:56    0.0
 8       65.1127           1.0     96 09:36:56    0.0

> summary(rcb.asr)$varcomp
             component   std.error   z.ratio bound %ch
Block!Block 0.06715427 0.068195046 0.9847383     P   0
units!R     1.00000000          NA        NA     F   0
units!units 0.05014295 0.007314089 6.8556664     P   0
```

Note the overall scale parameter `units(R)` is fixed at `1.0` and there is an estimated `units` variance. For both correlation (gamma parameterization) and variance (sigma parameterization) models for the residual, ASReml-R automatically includes an overall scale parameter. When a variance model (with associated variance parameter) is specified, the overall scale parameter is fixed at 1.0 to avoid overparameterization. This is reflected by the constraint on `units(R)` in the summary table.

3.1 Specification of the linear mixed model

Using `gammaPar=TRUE` in `asreml.options()`

For single section models where the residual formula specifies a variance model with a single parameter, the default action to use the sigma parameterization can be switched to the gamma parameterization by setting `asreml.options(gammaPar=TRUE)`:

```
asreml.options(gammaPar = TRUE)
rcb.asr <- asreml(fixed = yield~1 + Variety, random = ~ idv(Block),
+ residual = ~idv(units), data = rcb.dat)

Model fitted using the gamma parameterization.
ASReml 4.1.0 Fri Feb  2 09:43:13 2018
          LogLik        Sigma2     DF     wall    cpu
 1       58.2386     0.0608567     96 09:43:13    0.0
 2       60.4027     0.0578118     96 09:43:13    0.0
 3       62.6539     0.0546469     96 09:43:13    0.0
 4       64.1290     0.0524343     96 09:43:13    0.0
 5       64.8815     0.0510398     96 09:43:13    0.0
 6       65.0836     0.0504189     96 09:43:13    0.0
 7       65.1117     0.0501896     96 09:43:13    0.0
 8       65.1127     0.0501448     96 09:43:13    0.0
```

Note the message to the screen indicating that the gamma parameterization has been used.

The reported variance parameters

By default, the sigma parameterization is used for reporting parameters:

```
summary(rcb.asr)$varcomp
             component  std.error   z.ratio bound %ch
Block!Block 0.06715656 0.06802034 0.9873012     P 0.2
units(R)    0.05014482 0.00731451 6.8555269     P 0.0
```

Variance ratios estimated using the gamma parameterization can be reported by setting the `param` argument of `summary.asreml()` to `"gamma"`:

```
summary(rcb.asr, param = "gamma")$varcomp
              gamma  component  std.error   z.ratio bound %ch
Block!Block 1.339252 0.06715656 0.06802034 0.9873012     P 0.2
units(R)    1.000000 0.05014482 0.00731451 6.8555269     P 0.0
```

It can sometimes be advantageous to switch to the gamma parameterization in terms of providing more appropriate initial (starting) values as can be seen by comparison of the log-likelihoods in the first iteration. We can see that for the simple RCB example the REML log-likelihood is the same for the two fits (sigma then gamma parameterization) and the REML estimates of the two variance parameters are also identical.

The gamma parameterization by default

To ensure upward compatibility with previous releases, ASReml-R also uses the gamma parameterization for model fitting by default if either no residual formula is specified or the residual formula specifies a correlation structure. For example:

```
residual = ~id(units)
residual = ~ar1(Column):ar1(Row)
residual = ~id(units):cor(trait)
```

would all use the gamma parameterisation. The following is the likelihood convergence and default table of variance parameter estimates when an ID variance structure is specified for the residual model:

```
rcb.asr <- asreml(fixed = yield~1 + Variety, random = ~ idv(Block),
residual = ~id(units), data = rcb.dat)

Model fitted using the gamma parameterization.
ASReml 4.1.0 Fri Feb  2 09:45:55 2018
          LogLik        Sigma2     DF     wall    cpu
 1       58.2386     0.0608567     96 09:45:55    0.0
 2       60.4027     0.0578118     96 09:45:55    0.0
 3       62.6539     0.0546469     96 09:45:55    0.0
 4       64.1290     0.0524343     96 09:45:55    0.0
 5       64.8815     0.0510398     96 09:45:55    0.0
 6       65.0836     0.0504189     96 09:45:55    0.0
 7       65.1117     0.0501896     96 09:45:55    0.0
 8       65.1127     0.0501448     96 09:45:55    0.0

> summary(rcb.asr)$varcomp
             component   std.error   z.ratio bound %ch
Block!Block 0.06715656 0.068019908 0.9873075     P 0.2
units!R     0.05014483 0.007314511 6.8555266     P 0.0
```

Multi-section models using dsum

In the case of multi-section models where the variance structure for the residual error term is a direct sum (specified using dsum) the sigma parameterization is used, see Section 4.2 for more detail.

3.2 asreml() arguments

`family` The `family` argument can optionally accept a list of family functions for bi-variate GLM models.

GLM `families` The GLM family functions (gaussian, binomial, etc) are now prefixed by `"asr_"` instead of the Version 3 style `"asreml."` prefix. The complete set of families includes `gaussian`, `Gamma`, `inverse.gaussian`, `binomial`, `multinomial`, `negative.binomial`, `poisson`. A bivariate example is:

```
binnor.asr <- asreml(cbind(score5, norm) ~ trait:Sex + trait:Grp,
                     random = ~ us(trait):Sire,
                     family=list(asr_binomial(),asr_gaussian()),
                     data=binnor, maxit=20)
```

A multinomial example is discussed in Section 4.1.

`knot.points` User supplied knot points for spline terms. `knot.points` in Version 4 replaces

splinepoints in Version 3. For example:

```
asreml.options(predictpoints = list(x = seq(118,1582,by=1)))
orange.asr <- asreml(circ $$ x, random =  spl(x),
 knot.points = list(x = c(118,484,664,1004,1231,1372,1582)), data = orange, subset = Tree==1)
```

model.frame The model.frame argument optionally accepts a text string naming an RDS file (type ?readRDS in R for information on RDS files) in which to store the asreml model frame (data + model attributes) outside the asreml object. The motivation is to reduce the size of the asreml object in large examples. The data as used in the fit is needed for some post fitting methods like resid and plot, but automatically including it in the object adds to the size of the object and the .RData file, particularly if the data alone is hundreds or thousands of megabytes. The RDS file is a convenient way to keep the data (model) frame and all its properties outside the .RData workspace. As an example, in

```
oats.asr <- asreml(yield ~ variety*nitrogen,....,data=oats, model.frame="oats.RDS")
```

oats.RDS contains the data used in the analysis (i.e. after any model functions etc. have had their way) as a data.table object with numerous internal attributes. The model.frame component of oats.asr then just contains the string oats.RDS, which then allows plot() to find it.

mbf Component name cov replaces dataFrame to identify the covariate dataframe. For example:

```
naf.asr <- asreml(yield ~ type,
           random = ~ mbf(A)+Variety, data=naf,
           residual = ~ ar1(column):ar1(row),
           mbf = list(A=list(key=c("Variety","V1"),cov="naf31H")))
```

fits a function associated with factor A defined in dataframe naf31H.

na.action Specifies the action to be taken when missing values are encountered in the response or explanatory variables. In the call, na.action = na.method(). The arguments to na.method() are y (the response) = "include", "omit" or "fail" and x (explanatory variables) = "fail", "include" or "omit". The default action is to include (and estimate) missing values in the response (y = "include") and to raise an error if there are missing values in the explanatory variables (x = "fail").

pwr.points User supplied distances for one-dimensional power models (replaces pwrpoints). For example:

```
asreml(y ~ 1, random = ~ expv(time) + ..., pwr.points=list(time=c(2,4,7,12))
```

If not given, expv() gets the points from unique(data[[time]]).

workspace Memory settings (workspace and pworkspace) are now in the options list, but can also be specified in the asreml() function call. These may be set with text strings specifying the memory units, for example "3gb" requests 3 gigabytes of space. Valid units are "kb", "mb" and "gb" which are not case sensitive, for example, "Kb", "kB", "KB" and "kb" refer

identically to kilobytes.

Note that 100e6 in Version 3 is 8*100e6 bytes which is approximately `workspace = "800mb"` in `Version 4`. Note that, no suffix still gives the Version 3 workspace calculation in double precision words (= `8 bytes`).

3.3 The `asreml` object

`Cfixed` Returned as a `dspMatrix` class matrix. A `dspMatrix` class object is a dense symmetric matrix provided by the `Matrix` sparse matrix package in R. Only $n(n+1)/2$ elements are stored but are visible to the user through normal subscripting. The `Matrix` package is installed by default in R.

`ai` Returned as a `dspMatrix` class matrix. See `Cfixed` for information on `dspMatrix` class objects.

3.4 Screen output

ASReml-R no longer prints `LogLikelihood Converged` and `LogLikelihood not converged` messages to the screen at the end of a run. All warnings and failures are reported. A return to the prompt with no warnings/failures indicates the model has converged.

3.5 Mixed model functions

`str()` The function to specify a general variance structure spanning consecutive model terms has been updated. In particular, the specification of variance models that generate new model factors with a modified set of levels (fa(), ped() and ide() in Version 3, fa(), rr(), vm() and ide() in Version 4) has changed. See Section 4.3.7 of the ASReml-R Reference Manual Version 4 for details.

3.6 Methods and functions

`asreml.options()` Less frequently used settings are now set per session outside the `asreml()` function call in an *options environment* by a call to `asreml.options()`. Settings that were formerly set in `asreml.control` but are used more often, for example, `mbf` and `group`, are now set in the call to `asreml()`. With no arguments, `asreml.options()` returns a list of settings that can be altered; arguments are a sequence of `name=value` pairs. The full list of options is given in the ASReml-R package reference available at http://asreml.org under `Resources` > `ASReml docs` and by typing `help(asreml)` in R. The options that have changed are as follows:

`ai.penalty` The algorithm for updating loadings in factor analytic models has been improved. The motivation for change was that the original update procedure sometimes produced unreasonable updates, or otherwise came near to convergence and then drifted away. The present procedure is to modify 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, ASReml-R

will increase the value of p by 10% and then continue. The initial value of p is set in the `ai.penalty` component of `asreml.options()` list. After the penalty has reduced to 1%, it is further reduced to 0.2%. The value can be set to 0 if desired.

```
asreml.options(ai.penalty = 10) #the default
asreml.options(ai.penalty = 0) #no modification
```

`design` The `design` argument to `asreml.options` takes values `TRUE` or `FALSE` (default). If `TRUE`, the design matrix is returned in component `design` of the `asreml` object. The `design` option might be used, for example, in generating design matrices that can then be used in post-processing eg. to check design matrices.

`font.scale` Scales axis text and labels (relative to the ASReml-R default settings) in the graphs generated by `plot.asreml()`. The default is `font.scale=1.0`.

`spline.scale` Replaces `splinescale` which was formerly set in `asreml.control`.

`trace` If `TRUE` then a component called `trace` is added to the returned `asreml` object containing information regarding the convergence of the current fit.

`ainverse()` Renamed from `asreml.Ainverse()`. The returned object is now a three-column matrix of class `ginv` with attributes `rowNames`, `inBreeding`, `geneticGroups` and `logdet` (previously a list of these objects in Version 3).

`na.method()` See `na.action`, Section 3.2.

`plot.asreml()` Now uses `ggplot2` graphics.

`predict.asreml()` The following are changes to `predict.asreml()` with Version 4:

- a new `design.points` argument, formerly known as `predictpoints` in `asreml.control()`
- field name `standard.error` in the predictions data frame changed to `std.error`
- covariances and standard errors returned as `dspMatrix` objects, see `Cfixed` above for information on `dspMatrix` class objects
- `predict()` just returns the `predictions` component of the `asreml` object.

`summary.asreml()` Variance parameters are reported using the sigma parameterisation by default. Use `param="gamma"` to also report gamma parameters. The following are changes to `summary.asreml()` with Version 4:

- the `nice` argument to `summary.asreml()` has been renamed `vparameters` and the `nice` component of the summary object is now the `vparameters` component.
- the `all` argument to `summary.asreml()` has been renamed `coef`. Setting `coef=TRUE` allows the coefficients for the fixed, random and sparse model terms and their standard errors to be accessed through the `coef.fixed`, `coef.random` and `coef.sparse` components, respectively.
- the `constraint` field in the `varcomp` component has been renamed `bound`.

For comparing nested models we recommend the REML likelihood ratio test, see Section 2.4.1 of the ASReml-R Reference Manual Version 4. The *Akaike Information Criterion* (AIC) and the *Bayesian Information Criterion* (BIC) are also defined in Section 2.4.1 of the reference manual. The AIC and BIC are provided for the convenience of users but without any formal recommendation for their use. The number of parameters includes the number of linear parameters estimated and the number of variance structure parameters estimated, excluding variance parameters fixed at a boundary during the estimation procedure. The value used in calculating AIC and BIC is reported, giving the opportunity for the user to verify/modify this number.

All of these statistics are based on the REML log-likelihood statistics and are only valid if the fixed effects model is unchanged between runs and is fitted in the same order (ie. the same effects are aliased in the case where the model is over-parameterized).

`variogram()` Renamed to `varioGram()` to avoid conflict with similar naming in other S packages; method renamed to `varioGram.asreml()` and the variogram constructor renamed to `asr_varioGram()`.

3.7 Lists

`formulae` A list with elements the fixed, random, sparse and residual model formulae specified in the call to `asreml()`.

4 What's new in ASReml-R Version 4

4.1 `asreml()` arguments

`ai.loadings` This option allows further control of the AI updates of loadings in extended factor-analytic (fa(,*k*)) models. After ASReml-R 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. Further, when ai.loadings=*n* is specified (default $n = -1$ specifies no action) and the user has not imposed identifiability constraints, then ASReml-R imposes them using ai.rotate=TRUE and it also prevents AI updates of some loadings during the first n iterations. For $k > 1$ factors, only the last factor is estimated (conditional on the earlier ones) in the first $k-1$ iterations. Then pairs, including the last, are estimated until iteration n.

`asr_multinomial` The multinomial family is new in Version 4 and allows the fitting of multiple threshold models to polytomous ordinal data with k categories with $t = k - 1$ thresholds, assuming a multinomial distribution. Typically, the response variables are a set of k variables containing counts r_i $(i = 1 \ldots k)$ in the k categories. 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. The total, $n = \sum_{i=1}^{k} r_i$ say, can be given using the argument `total=n`. If the response is a matrix then the data is grouped and if `total=NULL`, the total counts are calculated from the category columns. If the response is a vector then the data are un-grouped and the total is not relevant. In this case the response must be a factor (this is the only case where ASReml-R allows a response variable to be a factor) and the response factor must have at least 2 levels.

The model is fitted by transforming the counts to proportions $y_i = r_i/n$, $(i = 1 \ldots k)$ and forming cumulative proportions $Y_i = \sum_{j=1}^{i} y_j$ and modelling $E(Y_i)$ with μ_i and $\nu_i = h(\mu_i)$ with ν_i the linear predictor for the ith variable and $h()$ the link function. Typically the linear predictors have two parts, one the same for all the variables associated with one record and a second being a threshold differing for each variable. These thresholds can be specified using the trait variable. As an example, the data `cheese` contains counts on four cheeses in 9 categories. We wish to model the data using a logistic distribution (the default distribution) with 8 $(= 9 - 1)$ thresholds to predict the probabilities in each category and to allow different effects for each cheese.

A specification of this multinomial model in vectorised form is:

```
ch4.asr <- asreml(Taste ~ trait + Cheese,
                  residual = ~ units:mthr(trait),
                  family=asr_multinomial(),data=cheese.cat)
```

In grouped form we have:

```
## total not given
ch1.asr <- asreml(cbind(cat1,cat2,cat3,cat4,cat5,cat6,cat7,cat8,cat9) ~ trait + Cheese,
                  family=asr_multinomial(),data=cheese)

## total in "tot"
ch2.asr <- asreml(cbind(cat1,cat2,cat3,cat4,cat5,cat6,cat7,cat8,cat9) ~ trait + Cheese,
                   family=asr_multinomial(total=tot),data=cheese)
```

`combine` Form a new factor from an existing factor by merging a subset of its levels, see also `gpf()` (4.2). For example, for a factor `Site` with three levels `Site1`, `Site2` and `Site3`, the following code within an `asreml()` function call would create a new 2 level factor `C` with level 1 being sites 1 and 3 and level 2 being site 2:

```
combine=list(C=Levels(Site, c('1','2','1')))
```

This subset factor `C` can be incorporated into model formulae using `gpf`, eg. `gpf(C)` or `idv(gpf(C))`.

`dense` ASReml-R uses linked-list matrix methods for the sparse equations, that is, equations for terms in the `random` and `sparse` formulae. However, genetic relationship matrices, for example, are typically quite large (order several thousand) and dense, and it can be more efficient to process such terms as dense. The `dense` formula component of the `asreml.options()` list can be used to include terms in the `random` formula in the dense set of equations for processing. The `dense` formula can also be given directly as an argument to `asreml()`. For example:

```
asreml.options(dense = ~ vm(clonefv,K))
nassau.asr <- asreml(ht6 ~ CultureID/Rep,
                     random = ~ vm(clonefv,K) + ide(clonefv) + Rep:IncBlock ...)
```

or equivalently:

```
nassau.asr <- asreml(ht6 ~ CultureID/Rep,
                     random = ~ vm(clonefv,K) + ide(clonefv) + Rep:IncBlock,
                     dense = ~ vm(clonefv,K))
```

would include the equations for `clonefv` in the dense set for processing.

`family` GLM families include the `asr_multinomial()` function.

`gammaPar` Allows the user to change between the sigma and gamma parameterizations for model fitting. If `gammaPar=FALSE` within `asreml.options()`, the parameterization used depends on the specification of the model for the residual error term. The sigma parameterization

is used if the residual model is specified as a variance matrix or with `dsum`. The gamma parameterization is used if the residual model is specified as a correlation matrix. There are sometimes advantages in using a gamma parameterization in terms of providing appropriate initial values. If the residual model has only one variance parameter then setting `gammaPar=TRUE` within `asreml.options()` switches from the sigma parameterization to the gamma parameterization in fitting the mixed model.

`rotate.fa` Constraints are required in $\mathbf{\Gamma}$ for $k > 1$ for identifiability in fa(,k) models. These are automatically set unless the user ensures identifiability by constraining one parameter in the second column, two in the third column, etc. With rotate.fa=FALSE (the default), ASReml-R fixes the $j = 1, ..., i-1$ loadings for the i-th factor ($2 \leq i \leq k$) to zero and their corresponding boundary constraints to F. The total number of constraints is $k(k-1)/2$.

An alternative set of constraints can be set if identifiability constraints have not been imposed, using rotate.fa=TRUE. The factors are rotated to orthogonality, in each iteration, and $k(k-1)/2$ constraints are imposed on the loadings depending on the values in this orthogonalized $\mathbf{\Gamma}$. This option is hypothesized to have better convergence properties but we do not have sufficient evidence yet to make a definite recommendation on its use. We note that extra constraints might be needed to ensure identifiability if the number of parameters in a k factor analytic model, $\omega(1+k) - k(k-1)/2$, is greater than the $\omega(\omega+1)/2$ parameters that can be estimated in $\mathbf{\Sigma}$.

`prune` Form a new factor from an existing factor by selecting a subset of its levels, see also `sbs()` (4.2). For example, for the same factor as in `combine`, the following code within an `asreml()` function call would instruct ASReml-R to form factors `A` and `B` with only those levels in the specified subset of `Site`:

```
prune=list(A=Subset(Site, c(2,3)), B=Subset(Site,c("Site1","Site3")))
```

This example demonstrates the two ways of specifying the required levels of site, ie. by level number for factor `A` and by the site name for factor `B`. The factors `A` and `B` can be used in the model term using `sbs`, for example, `sbs(A)` to fit a fixed effect or `idv(sbs(B))` to fit a variance component.

`vcc` Is the command that allows users the functionality of the `constraints` argument in Version 3.

Equality and multiplicative relationships among variance parameters are defined by supplying a two-column numeric matrix with a dimnames attribute to `vcc`. The first column defines the grouping of variance parameters by assigning the same number to each parameter within a group, 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 in a group are scaled relative to the first parameter in that group so that the scaling of the first parameter in each group is one.

For example, consider a MET with two trials with separate error variances (σ_1^2 and σ_2^2) and the spatial row (ρ_{r_1} and ρ_{r_2}) and column (ρ_{c_1} and ρ_{c_2}) parameters for a separable autoregressive spatial model of order 1 for each trial. Say we wish to constrain these error models to be

equal so that $\sigma_1^2 = \sigma_2^2$, $\rho_{r_1} = \rho_{r_2}$ and $\rho_{c_1} = \rho_{c_2}$. Then the appropriate vcc matrix with row attributes is

$$\begin{array}{ll} \texttt{Trial\_1!R} & \left[\begin{array}{cc} 1 & 1 \\ 2 & 1 \\ 3 & 1 \\ 1 & 1 \\ 2 & 1 \\ 3 & 1 \end{array}\right] \\ \texttt{Trial\_1!Row!cor} & \\ \texttt{Trial\_1!Column!cor} & \\ \texttt{Trial\_2!R} & \\ \texttt{Trial\_2!Row!cor} & \\ \texttt{Trial\_2!Column!cor} & \end{array}$$

Alternatively, if we require $\sigma_2^2 = 2\sigma_1^2$, the vcc matrix is

$$\begin{array}{ll} \texttt{Trial\_1!R} & \left[\begin{array}{cc} 1 & 1 \\ 2 & 1 \\ 3 & 1 \\ 1 & 2 \\ 2 & 1 \\ 3 & 1 \end{array}\right] \\ \texttt{Trial\_1!Row!cor} & \\ \texttt{Trial\_1!Column!cor} & \\ \texttt{Trial\_2!R} & \\ \texttt{Trial\_2!Row!cor} & \\ \texttt{Trial\_2!Column!cor} & \end{array}$$

`vcm` Allows the user to define equality and multiplicative relationships among variance parameters. The default `NULL` means no relationship is fitted.

The `vcm` argument to asreml() allows the user to define equality and multiplicative relationships among variance parameters. The default `NULL` means no relationship is fitted.

The user may wish to define relationships between particular variance parameters. For example, consider an experiment in which two or more separate trials are sown adjacent to one another at the same trial site, with trials sharing a common plot boundary. In this case it might be sensible to fit the same spatial parameters and error variances for each trial. In other situations it can be sensible to define the same variance structure over several model terms. ASReml-R Version 3 catered for equality and multiplicative relationships among variance parameters (this facility is available in Version 4 through vcc, see above). In ASReml-R Version 4 linear relationships among variance structure parameters can be defined through a simple linear model and by supplying a design matrix for a set of parameters.

Let $\boldsymbol{\kappa}$ be the r-vector of original variance parameters (for either the sigma or gamma parameterisation) from which we wish to specify linear relationships of the form

$$\boldsymbol{\kappa} \quad = \quad \boldsymbol{M}\boldsymbol{\kappa}_n$$

where $\boldsymbol{\kappa}_n$ is the c-vector of parameters in the new set. In the simple case where the r parameters are constrained to be equal, $c = 1$, the r original parameters are all equal to the one new parameter and $\boldsymbol{M}$ will contain a column of ones. Consider again the MET with two trials in which we wish to constrain the trial error variances and the spatial row and column parameters for a separable autoregressive spatial model of order 1 for each trial, to be equal. In this case the relationship between the original and new parameter sets is $\boldsymbol{\kappa} = \boldsymbol{M}\boldsymbol{\kappa}_n$ where $\boldsymbol{\kappa}$ is the 6×1 vector $[\sigma_1^2,\ \rho_{r_1},\ \rho_{c_1},\ \sigma_2^2,\ \rho_{r_2},\ \rho_{c_2}]^\mathsf{T}$, $\boldsymbol{\kappa}_n$ is a 3×1 vector $[\sigma_e^2,\ \rho_r,\ \rho_c]^\mathsf{T}$ and $\boldsymbol{M}$,

with row attributes, is the 6×3 matrix

$$\begin{array}{ll} \texttt{Trial\_1!R} & \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \\ \texttt{Trial\_1!Row!cor} & \\ \texttt{Trial\_1!Column!cor} & \\ \texttt{Trial\_2!R} & \\ \texttt{Trial\_2!Row!cor} & \\ \texttt{Trial\_2!Column!cor} & \end{array}$$

Generating M

A default data frame vparameters.table is generated by setting the start.values argument to TRUE in the call to asreml(). This data frame contains elements Component which contains the names of the variance parameters, Value which contains the default initial values and Constraint which contains the default constraint code. The Component element can be used to generate $\boldsymbol{M}$.

By way of example, consider a model containing a first order interaction term ($\boldsymbol{A} : \boldsymbol{B}$, say) where the outer factor ($\boldsymbol{A}$) is of order 7 and we wish to model it with an unstructured variance matrix with some parameters constrained. If the constraints are
$v_{r,c} = v_{3,c}$ $(r = 4,5,6,7;\ c = 1,2)$, $v_{r,r} = v_{3,3}$ $(r = 4,5,6,7)$ and $v_{r,c} = v_{4,3} = 0$ $(r = 5,6,7;\ c = 3,4,5,6;\ r > c)$, this gives rise to a vector of 7 parameters
$\kappa_n = (v_{1,1}, v_{2,1}, v_{2,2}, v_{3,1}, v_{3,2}, v_{3,3}, v_{4,2})^{\mathsf{T}}$ and a variance matrix:

$$\begin{array}{ccccccc} v_{1,1} & & & & & & \\ v_{2,1} & v_{2,2} & & & & & \\ v_{3,1} & v_{3,2} & v_{3,3} & & & & \\ v_{3,1} & v_{3,2} & v_{4,3} & v_{3,3} & & & \\ v_{3,1} & v_{3,2} & v_{4,3} & v_{4,3} & v_{3,3} & & \\ v_{3,1} & v_{3,2} & v_{4,3} & v_{4,3} & v_{4,3} & v_{3,3} & \\ v_{3,1} & v_{3,2} & v_{4,3} & v_{4,3} & v_{4,3} & v_{4,3} & v_{3,3} \end{array}$$

That is, there are only 7 distinct parameters from the original 28 and one of these is to be fixed at zero. Furthermore, suppose that none of the remaining variance parameters from other terms in the model are to be subject to any constraints.

The following call

```
> model.gam <- asreml(..., random = us(A):id(B), start.values = T, ...)
```

generates a data frame component of model.gam named vparameters.table, as described above.

If the 28 components of interest are the 47^{th} to the 74^{th}, the following code subsets model.gam$vparameters.table and creates a factor in the reduced table that can be used to construct $\boldsymbol{M}$:

```
gam <- model.gam$vparameters.table[47:74,]
gam$fac <- factor(c(
```

```
                    1,
                    2,3,
                    4,5,6,
                    4,5,7,6,
                    4,5,7,7,6,
                    4,5,7,7,7,6,
                    4,5,7,7,7,7,6))
M <- model.matrix(~-1 + fac, data=gam)
dimnames(M)[[1]] <- row.names(model.gam$vparameters.table)[47:74]
attr(M,'assign') <- NULL; attr(M,'contrasts') <- NULL
```

Important $\boldsymbol{M}$ must have a dimnames attribute with the names of the original set of parameters as its row names.

In this example, $\boldsymbol{M}$ would look like

parameter	attribute	κ_n	1	2	3	4	5	6	7
47	A:B!B_1!1	$v_{1,1}$	1	0	0	0	0	0	0
48	A:B!B_2!1	$v_{2,1}$	0	1	0	0	0	0	0
49	A:B!B_2!2	$v_{2,2}$	0	0	1	0	0	0	0
50	A:B!B_3!1	$v_{3,1}$	0	0	0	1	0	0	0
51	A:B!B_3!2	$v_{3,2}$	0	0	0	0	1	0	0
52	A:B!B_3!3	$v_{3,3}$	0	0	0	0	0	1	0
53	A:B!B_4!1	$v_{4,3}$	0	0	0	1	0	0	0
54	A:B!B_4!2		0	0	0	0	1	0	0
55	A:B!B_4!3		0	0	0	0	0	0	1
56	A:B!B_4!4		0	0	0	0	0	1	0
57	A:B!B_5!1		0	0	0	1	0	0	0
58	A:B!B_5!2		0	0	0	0	1	0	0
59	A:B!B_5!3		0	0	0	0	0	0	1
60	A:B!B_5!4		0	0	0	0	0	0	1
61	A:B!B_5!5		0	0	0	0	0	1	0
62	A:B!B_6!1		0	0	0	1	0	0	0
63	A:B!B_6!2		0	0	0	0	1	0	0
64	A:B!B_6!3		0	0	0	0	0	0	1
65	A:B!B_6!4		0	0	0	0	0	0	1
66	A:B!B_6!5		0	0	0	0	0	0	1
67	A:B!B_6!6		0	0	0	0	0	1	0
68	A:B!B_7!1		0	0	0	1	0	0	0
69	A:B!B_7!2		0	0	0	0	1	0	0
70	A:B!B_7!3		0	0	0	0	0	0	1
71	A:B!B_7!4		0	0	0	0	0	0	1
72	A:B!B_7!5		0	0	0	0	0	0	1
73	A:B!B_7!6		0	0	0	0	0	0	1
74	A:B!B_7!7		0	0	0	0	0	1	0

The final step before fitting the model is to fix the parameters corresponding to level 7 of fac to zero. This is achieved by by setting the appropriate values in the Value field of gam

to zero and the corresponding boundary constraint codes in the Constraint field to F. During the estimation procedure the new parameters, κ_n, use the numbering system of the original parameters, κ, hence the 7 κ_n parameters are numbered from 47-53. So to fix $v_{4,3}$, parameter 53 is fixed. The modified values and the matrix $\boldsymbol{M}$ are used through the G.param and vcm() arguments to asreml(), that is

```
model.asr <- asreml(..., random = us(A):id(B), vcm(M), G.param = gam, ...)
```

This example has been set up to show how vcm() can be used. An equivalent method in this case would be to fix the 10 parameters $v_{r,c}$ ($r = 4, 5, 6, 7$; $c = 3, 4, 5, 6$; numbers 55, 59, 60, 64, 65, 66, 70, 71, 72, 73) and set up a 15×3 matrix based on parameters $v_{r,c}$ ($r = 3, 4, 5, 6, 7$; $c = 1, 2$; numbers 50, 51, 53, 54, 57, 58, 62, 63, 68, 69) and $v_{r,r}$ ($r = 3, 4, 5, 6, 7$; numbers 52, 56, 61, 67, 74) in terms of $v_{3,1}$, $v_{3,2}$ and $v_{3,3}$.

Warning: Beta-testing of Version 4 has indicated that if the modelled parameters $\boldsymbol{\kappa} = \boldsymbol{M}\boldsymbol{\kappa}_n$ need to be restrained, the variance parameters are not at their optimum values. Please ask VSN International for further details on how to overcome this challenge.

4.2 Mixed model functions

`C()` Factor contrasts. In the following `L` is a contrast based on the 4 levels of a factor `nitrogen`:

```
L <- c(3,1,-1,-3)
```

The contrast would be defined by the term `C(nitrogen, L)` in the model call.

The length of the contrast vector (pre-defined as `L` in this case although `c(3,1,-1,-3)` could replace `L` in the call) must be equal to the number of levels in the factor. Missing values in the factor are excluded in forming the contrast.

`dsum()` Direct sum structures for the residual error term

The data observations are often partitioned into sections to which separate variance structures are applied. For example, separate spatial structures and residual error variances would typically be specified for each site in a multi-environment trial (`MET`) analysis.

It is conventional to use a variable in the data file to identify sections within the data. The data will be sorted internally by ASReml-R (ie. the data file does not need to be ordered in any particular way) and the variance structures for sections can then be specified using the `dsum` function, for example:

```
residual = ~dsum(~id(units) | section)
```

for a simple analysis in which `section` is a column in the data file that codes the individual sections. The `dsum` function (shorthand for *direct sum*) is new with Version 4 and performs several different tasks:

- it tells ASReml-R that the variance structure for the residual error term is a direct sum structure where different variance structures apply to the different levels of the sectioning variable in the data.

  If a model structure specified defines a residual matrix then a variance factor associated with the appropriate sectioning level is added to the specified model to generate a variance matrix.

- it prunes the levels for a section so that *only the levels of factors defining the residual variance structure for that section are used in forming that variance structure.*

Variance model functions in `dsum`

Correlation models were used in direct sum structures for the residual error term in Version 3 which automatically added and estimated a scale parameter for each section. In Version 4 a variance model function can be specified for one argument of the `dsum` component for each section. In this case the section variance is automatically fixed at 1.0 to avoid over-parameterization. For each section, ASReml-R counts the number of dimensions (1 for a single term, ≥ 2 for separable structures) for which variance models are specified and if the count is > 1 the model is judged to be over-parameterized and an error is returned.

Specifying the model using `dsum`

Often sections relate to sites (or trials or experiments) in the case where several related trials are analysed together. For example, consider a MET dataset comprising data for three sites, each laid out in a row by column array coded by factors `Row` and `Column` in the dataset. To model the residuals at each site by a separate scaled AR1×AR1 variance structure, we could write:

```
residual = ~dsum(~ar1(Column):ar1(Row) | Site)
```

Alternatively, a scaled AR1×AR1 variance structure for sites 1 and 3, but a scaled ID×AR1 structure for site 2, could be coded as:

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

or as

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

where `Site1`, `Site2` and `Site3` are the three site labels. An alternative is to provide separate `dsum` statements for the AR1×AR1 and ID×AR1 sections:

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

Making use of variance model functions in `dsum`, other variants on this code are:

```
residual = ~dsum(~ar1v(Column):ar1(Row) + idv(Column):ar1(Row) | Site,
levels = list(c(1,3), c(2)))
```

and

```
residual = ~dsum(~ar1(Column):ar1v(Row) + id(Column):ar1(Row) | Site,
levels = list(c(1,3), c(2)))
```

For the former, the error variance would be fixed at `1.0` for all three sites to avoid overparameterization. For the latter, the error variances for sites 1 and 3 but not 2 would be fixed at `1.0`. An error would be returned for

```
residual = ~dsum(~ar1v(Column):ar1v(Row) + id(Column):ar1v(Row) | Site,
levels = list(c(1,3), c(2)))
```

```
Error:  Residual model overparameterized - structure has 2 variance models
```

For each of these definitions, ASReml-R will determine the particular levels in `Row` and `Column` for each site and hence the appropriate sizes of the AR1 and ID matrices, and variances associated with the levels of `Site` will be added to correlation structures.

Important A correlation/variance structure needs to be specified for every level of the sectioning factor, in which case

```
residual = ~dsum(~ar1(Column):ar1(Row) | Site, levels=c(1,3))
```

would fail as there is no variance structure specified for site 2.

Specifying variance structures using dimensions

Although less conventional, variance structures can also be specified using dimensions rather than factor names. For example, consider a simple MET comprising three trials arranged in rectangular arrays of dimension 12, 10 and 9 rows by 6, 8 and 18 ranges for trials 1, 2 and 3, respectively. For data ordered rows within columns within trials (trials coded as `Site` in the data frame), an AR1×AR1 variance structure for trials 1 and 3 and an IDV×AR1 structure for trial 2, could be coded as:

```
residual = ~dsum(~ar1(6):ar1(12)| Site, levels=c(1)) + dsum(~ar1(8):ar1(10)| Site, levels=c(2))
    + dsum(~ar1(18):ar1(9)| Site, levels=c(3))
```

The `outer` argument to `dsum`

The `outer` argument to `dsum` has been introduced to enable modelling multiple independent sections of correlated observations with a common variance structure and common parameters within sections. The sections can be of different sizes. For example:

```
residual=~ dsum(~id(Range):ar1(Row)| Site,levels=c(1:2,7)) +
           dsum(~id(Range):ar1(Row)| Site,levels=3:4, outer=T) +
           dsum(~id(Range):ar1(Row)| Site,levels=5:6, outer=T),
```

would model separate error variance and spatial correlation parameters for levels 1, 2 and 7 of `Site` and the same error variance and spatial correlation parameters for levels 3 and 4 of the factor `Site` and likewise for levels 5 and 6 of `Site`.

Two rules for defining the residual error term

The following two rules are not new to ASReml-R with Version 4 but are included here as a reminder:

Rule 1 The number of effects in the residual term *must* be equal to the number of data units included in the analysis.

Rule 2 Where a separable variance structure is specified for the residual error term, each combination of levels of the single model terms specifying this structure must uniquely identify one unit of the data. For example, in the spatial analysis of a trial comprising 4 replicates of 24 varieties arranged as a rectangular array of dimension 4 rows by 24 columns (rows are replicates), a, AR1 × AR1 variance structure for the residuals can be specified by the model term `ar1(column):ar1(row)`, where `column` and `row` are the appropriate columns in the data file. However, the number of data units must be the product of the number of levels for `row` and the number of levels for `column`; 96 in this case. If this is not the case, or if more than one unit is associated with some row column combination, ASReml-R will return an error message and it will not be possible to use `ar1(column):ar1(row)` for residual error. If there are fewer than 96 units and each row-column combination present is associated with one unit, then the data would need to be augmented by completing (padding out) the full rectangular array allow an appropriate analysis.

These rules will always be satisfied for a single section of data defined either by default (ie. with no residual variance structure specified) or in terms of the `units` factor. However, a mismatch in both size and ordering is possible when either multiple sections are present (as in MET analysis) or when non-identity variance model functions are used.

`facv()` an alternative form of the factor analytic model fa(). If the size of the matrix modelled is much larger than the number of factors, the sparse or extended formulation fa() is usually computationally preferable.

`gpf()` Used in conjunction with `combine` (see Section 4.1) to form a new factor from an existing factor by grouping together levels, for example:

```
met.gpf <- asreml(yield ~ Site,
                  random= ~ Genotype + Genotype:Site + gpf(C):Row,
                  residual = ~ idv(units), data = met,
                  combine=list(C=Levels(Site, c('1','2','1'))))
```

`lvr()` a linear variance model.

`leg()` Legendre polynomials. Search for `legendre` in the ASReml-R forum (http://www.vsni.co.uk/forum/) for a discussion of legendre polynomials.

4.2 Mixed model functions

`own()` User defined variance models.

The `own()` variance model allows the specification of a user-defined variance structure. This feature requires a user provided R function (the default is `myowngdg()`) that returns a list containing the variance matrix and a full set of partial derivative matrices, one for each parameter. Before each iteration, ASReml-R calls `myowngdg()` passing basic information on the variance parameters given by the `own()` model term, and retrieves the returned list containing the variance matrix and its derivatives.

We use the data set `shf` (an agricultural field experiment arranged in a rectangular grid of plots) to illustrate the use of `own()`. `my.ar1()` is an example *myowngdg* function; it duplicates the AR1 variance structure.

Ignoring the design factors, the following simple analysis models the residuals with separable autoregressive processes in the `Row` and `Column` dimensions:

```
library(asreml)
shf.ar1 <- asreml(yield ~ Variety,
                  residual = ~ ar1(Row):ar1(Column), data=shf)
```

The following call (not run yet) replaces the intrinsic ASReml-R variance model `ar1()` with the user-defined function `my.ar1()` for the `Column` factor:

```
shf.ar1 <- asreml(yield ~ Variety,
                  residual = ~ ar1(Row):own(Column, "my.ar1", 0.1, "R"), data=shf)
```

where the arguments to `own()` are:

`Column`	The object in the data.
`"my.ar1"`	The name of the user defined function as a character string.
`0.1`	A vector of initial parameter values.
`"R"`	A character vector specifying the parameter type(s), in this case `"R"` designates the single parameter as a `correlation`.

`rr()` A reduced rank variant of the factor analytic variance model function `fa()`.

`sbs()` Used in conjunction with `prune` to form a new factor from an existing factor by selecting a subset of its levels, see example for `prune`.

`sfa()` a scaled version of the factor analytic model `fa()`.

`vm()` Known variance models. These may be genetic relationship matrices or their inverses if they exist. If an inverse, it must have an `INVERSE` attribute set to `TRUE`. For example, the inverse relationship matrix `harvey.ai` would need to have an inverse attribute set manually:

```
attr(harvey.ai, "INVERSE") <- TRUE
```

unless it was constructed from a pedigree file using `ainverse()` in which case an `"INVERSE"=TRUE` attribute would be automatically set:

```
harvey.ai <- ainverse(harvey.ped)
```

The `vm()` function includes known singular cases. For example:

```
# Pedigree file example
# A data set originally distributed by Walt Harvey
# Calf Sire Dam Line AgeOfDam Y1 Y2 Y3
# 101 Sire_1 0 1 3 192 390 224
# 102 Sire_1 0 1 3 154 403 265
# 103 Sire_1 0 1 4 185 432 241
# 104 Sire_1 0 1 4 183 457 225
harvey.ped <- read.table("harvey.ped", header=TRUE, as.is=TRUE)
harvey <- asr.read.table("harvey.dat", header=TRUE)

harvey.ai <- ainverse(harvey.ped)

adg0.asr <- asreml(y3 ~1, random = ~vm(Calf,harvey.ai),data=harvey)
```

4.3 Methods and functions

`meff()` Link a relationship matrix to the regressor variables.

One use of a relationship matrix is to allow more computationally efficient fitting of random regression models associating a vector $\boldsymbol{u}$ of p factor effects with a vector $\boldsymbol{v}$ of m regression effects through the model $\boldsymbol{u} = \boldsymbol{M}\boldsymbol{v}$, where the $p \times m$ matrix $\boldsymbol{M}$ contains m regressor variables for each of the p levels of the factor. If $m \gg p$, it is more computationally efficient to fit the model with $\boldsymbol{Z}\boldsymbol{u}$ and a variance structure for $\boldsymbol{u}$ based on $\boldsymbol{K} = \boldsymbol{M}\boldsymbol{M}'$, where $\boldsymbol{Z}$ is the design matrix linking observations to factor levels, than a model fitting the regressor effects directly. A common case of such a situation is in genomic studies when $\boldsymbol{u}$ represents genotype effects and $\boldsymbol{M}$ is the $p \times m$ matrix of genetic marker scores.

The matrix $\boldsymbol{K}$ is constructed externally to the `asreml()` function call and used in the analysis with the `vm()` model function. $\boldsymbol{K}$ must have a dimnames attribute giving the levels of the model term defined in `vm()`. The marker (or regressor) effects can be obtained from the `meff.asreml()` method. For example:

```
K <- M%*%t(M)
nassau.asr <- asreml(ht6 ~ CultureID/Rep,
                     random = ~ vm(clonefv,K) + ide(clonefv) +
                     Rep:IncBlock, ...)
nassau.mef <- meff(nassau.asr, mef=list(K="M"), effects = ~vm(clonefv, K))
```

fits such a model and estimates the marker effects given that the matrix $\boldsymbol{K}$ is in the R object `K` and the original $p \times m$ matrix of marker scores is in the R object `M`. The reason for quoting the name "`M`" is so that when R is passing the arguments through to the `meff` function it will not evaluate the object (which is typically large), as this will cause issues with memory and speed.

The `meff()` method is not to be confused with the `mef` argument to `asreml()` that accepts the return value of the `meff` method.

4.3 Methods and functions

`lrt()` Calculates the likelihood ratio test for fitted models.

`vpc.char()` Returns a vector of variance parameter constraint codes (`P`, `U`, `F`, `B`, ...) corresponding to the numeric values returned in the `vparameters.com` component of the `asreml` object, see example under `vpredict()`.

`vpredict()` Compute functions of variance components and their approximate standard errors.

Functions of variance components and their standard errors can be obtained from the `vpredict()` function. As the variance parameter names can sometimes be long or unwieldy, the variance parameters are represented in `vpredict()` by the strings `"V1", "V2",...` in the order in which they appear in the `vparameters` component of the `asreml` object. For example:

```
coop <- asreml.read.table("coop.dat", header=TRUE)
head(coop)
ywt0.sv <- asreml(cbind(ywt,fat) ~ trait + trait:age + trait:con(Brr) + trait:Sex + trait:Sex:age,
                  random = ~us(trait):id(Sire), sparse = ~trait:Grp,
                  residual = ~id(units):us(trait), data=coop, start.values=TRUE)

ywt0.sv <- ywt0.sv$vparameters.table
ywt0.sv[,'Value'] <- c(1.4, 0.13, 0.03, 1, 23, 2.5, 1.6)
ywt0.sv

ywt.asr <- asreml(cbind(ywt,fat) ~ trait + trait:age + trait:con(Brr) + trait:Sex + trait:Sex:age,
                  random = ~us(trait):id(Sire), sparse = ~trait:Grp,
                  residual = ~id(units):us(trait), data=coop, G.param=ywt0.sv, R.param=ywt0.sv)
ywt.asr$vparameters
#trait:Sire!trait_ywt:ywt  trait:Sire!trait_fat:ywt  trait:Sire!trait_fat:fat
#1.45821148                0.13027963                0.03443794
#units:trait(R) units:trait!trait_ywt:ywt units:trait!trait_fat:ywt
#1.00000000               23.20554057               2.50401740
#units:trait!trait_fat:fat
#1.66291555

#the variance parameter single character constraint codes
vpc.char(ywt.asr)
#trait:Sire!trait_ywt:ywt  trait:Sire!trait_fat:ywt  trait:Sire!trait_fat:fat
#                      "U"                       "U"                       "U"
#           units:trait(R) units:trait!trait_ywt:ywt units:trait!trait_fat:ywt
#                      "F"                       "U"                       "U"
#units:trait!trait_fat:fat
#                      "U"

ywt.vp <- cbind.data.frame(names(ywt.asr$vparameters),vpc.char(ywt.asr),
       1:length(ywt.asr$vparameters.type))
names(ywt.vp) <- c("names", "constraint", "number");ywt.vp
ywt.vp
#names                                          constraint number
#trait:Sire!trait_ywt:ywt   trait:Sire!trait_ywt:ywt          U      1
#trait:Sire!trait_fat:ywt   trait:Sire!trait_fat:ywt          U      2
#trait:Sire!trait_fat:fat   trait:Sire!trait_fat:fat          U      3
#units:trait(R)                       units:trait(R)          F      4
#units:trait!trait_ywt:ywt units:trait!trait_ywt:ywt          U      5
#units:trait!trait_fat:ywt units:trait!trait_fat:ywt          U      6
#units:trait!trait_fat:fat units:trait!trait_fat:fat          U      7
```

```
# heritA 4*V1 / (V1 + V5)
vpredict(ywt.asr, hA ~4*V1 / (V1 + V5))
#Estimate         SE
#hA 0.2364947 0.06117935

# heritB 4*V3 / (V3 + V7)
vpredict(ywt.asr, heritB ~4*V3 / (V3 + V7))
#Estimate         SE
#heritB 0.08115678 0.03936332

# genetic corr
vpredict(ywt.asr, gc ~V2 / sqrt((V1 * V3)))
#Estimate        SE
#gc 0.5813634 0.2038863

# phenotypic corr
vpredict(ywt.asr, pc ~(V2 + V6) / sqrt(((V1+V5) * (V3+V7))))
#Estimate         SE
#pc 0.4071449 0.01832069
```

Bibliography

Butler, D. G., Cullis, B. R., Gilmour, A. R. and Gogel, B. J. (2009). Asreml-r reference manual, version 3., *Queensland Department of Primary Industries* . Brisbane: Queensland Department of Primary Industries.