ASReml-R Reference Manual Version 4

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

D G Butler[1], B R Cullis[1], A R Gilmour[2], B J Gogel[3], and R Thompson[4]

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

[2]VSN International, formerly NSW Department of Primary Industries, Orange, Australia

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

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

August 12, 2019

ASReml-R Reference Manual Version 4

D. G. Butler, B. R. Cullis, A. R. Gilmour, B. J. Gogel 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., Cullis, B.R., A. R. Gilmour, Gogel, B.G. and Thompson, R. 2017. ASReml-R Reference Manual Version 4. VSN International Ltd, Hemel Hempstead, HP1 1ES, UK.

Author email addresses

dbutler@uow.edu.au
bcullis@uow.edu.au
Arthur.Gilmour@Cargovale.com.au
beverley.gogel@adelaide.edu.au
robin.thompson@rothamsted.ac.uk

Companion documents

Two main documents currently partner this reference manual. The document Navigating from ASReml-R Version 3 to 4 is a guide to transition existing users to the new version of ASReml-R and the ASReml-R package reference 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.

Acknowledgements

The authors gratefully acknowledge the Grains Research and Development Corporation of Australia for their financial support. We thank the University of Wollongong and Queensland Department of Agriculture and Fisheries for permitting this research to be undertaken and for providing a stimulating environment for applied biometrical consulting and research. We sincerely thank Sue Welham for her contribution toward research into the use of linear mixed models and the development of appropriate software. Ian White is acknowledged for suggesting a convenient way of computing approximate variances of functions of variance parameters, see the vignette Computing functions of variance parameters and their approximate standard errors available at http://www.homepages.ed.ac.uk/iwhite/asreml/uop.

Preface

ASReml-R is a statistical package that fits linear mixed models using Residual Maximum Likelihood (REML) in the R environment. This package uses the same computational kernel as its companion package ASReml. The computational kernel has been under development since 1993 and arose out of collaboration between Arthur Gilmour and Brian Cullis (NSW Department of Primary Industries) and Robin Thompson and Sue Welham (Rothamsted Research) to research into the analysis of mixed models and to develop appropriate software, building on their wide expertise in relevant areas including the development of methods that are both statistically and computationally effcient, the analysis of animal and plant breeding data, the analysis of spatial and longitudinal data and the production of widely used statistical software. Arthur Gilmour wrote the ASReml package. VSN International acquired the rights to the computational kernel and ASReml from these sponsoring organizations and now directly supports Arthur Gilmour for further computational developments. In parallel, David Butler and Brian Cullis (University of Wollongong) extended the computational kernel of ASReml to produce ASReml-R to work in the R environment. Beverley Gogel has been extensively involved in the documentation of both packages.

This reference manual documents the features of the methods for objects of class asreml. Outside of the worked examples, it does not consider the statistical issues involved in fitting models. The authors are contributing to the preparation of other documents that are focused on the statistical issues rather than the computing issues.

The features of ASReml-R include

- A flexible syntax for specifying variance models for the random effects, and the scope this offers the user. There is a potential cost for this complexity. Users should be aware of the dangers of either overfitting or attempting to fit inappropriate variance models to small or highly unbalanced data sets. We stress the importance of the use of data driven diagnostics and encourage the user to read the examples chapter, in which we have attempted to not only present the syntax of ASReml-R in the context of real analyses but also to indicate some of the modelling approaches we have found useful.

- The REML routines use the Average Information (AI) algorithm, and sparse matrix methods for fitting the mixed model. This enables ASReml-R to efficiently analyse large and complex data sets.

This manual consists of seven chapters. Chapter 1 introduces ASReml-R , describes the conventions used throughout the manual, and describes the various data sets used for illustration; Chapter 2 presents an general overview of basic theory; Chapter 3 presents an introduction to fitting models

in ASReml-R followed by a more detailed description of fitting the linear mixed model - this chapter includes a section (3.13) that describes the model specification for multivariate analyses; Chapter 4 is a key chapter that presents the syntax for specifying variance models for random effects in the model; Chapter 5 describes special functions and methods for genetic analyses; Chapter 6 outlines the prediction of linear functions of fixed and random effects in the linear mixed model and Chapter 7 presents a comprehensive and diverse set of worked examples. The ASReml-R Package Reference is a pdf version of the ASReml help pages obtained in R by typing `help(asreml)` and is available at `http://asreml.org` under `Resources` > `ASReml docs` and on the VSN International website `https://www.vsni.co.uk`. The Version 3 Reference Manual included the package reference as a chapter. For ease of updating information, the package reference is being kept as a separate document for 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. This functionality includes:

- 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 trail (MET) analysis.

- Simpler, more consistent specification and fitting of known variance matrices, including relationship matrices, and allowance for singular matrices.

- The own() variance model introduced 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 the use of derived model terms and functions.

- Functions that generate factors that combine levels of a factor or use a subset of levels to allow easier prediction of models.

These changes are summarized in a separate document Navigating from ASReml-R Version 3 to 4. A list of terms in Version 3 with their replacement terms in Version 4 is presented in Chapter 2 of the navigation document and is repeated in Appendix C. Chapters 3 and 4 of the navigation document

summarize what's changed and what's new in the Version 4 and are repeated in Appendices D and E.

The data sets and ASReml-R input files used in this manual are included in the software distribution. They remain the property of the authors or of the original source, but may be freely distributed provided the source is acknowledged. We have extensively tested the software but it is inevitable that bugs will exist. These may be reported to the authors. The authors would also appreciate being informed of errors and improvements to the manual and software.

Upgrades

ASReml-R is being continually upgraded to implement new developments in the application of linear mixed models. The release version will be available to licensed users from http://www.vsni.co.uk.

Contents

List of Tables

1 Introduction

1.1 What ASReml-R can do

ASReml-R is designed to fit the general linear mixed model to moderately large data sets with complex variance models. ASReml-R has application in the analysis of

- (un)balanced longitudinal data
- repeated measures data (multivariate analysis of variance and spline type models)
- (un)balanced designed experiments
- multi-environment trials and meta analysis
- regular or irregular spatial data.

The computational engine of ASReml-R is the algorithm of Gilmour et al. (1995) adapted from the standalone program ASReml (Gilmour et al.; 2002). The computational efficiency of ASReml-R arises from using this Average Information REML algorithm (giving quadratic convergence) and sparse matrix operations. However, because of overheads inherent in S language implementations, some very large problems may need to use the standalone ASReml program to overcome memory limitations.

The asreml() function returns an object of class asreml. Standard methods resid(), fitted(), coef(), summary(), plot(), anova() and predict() work with this object, and other methods including varioGram() and wald() also exist.

1.2 Getting started

1.2.1 Installation

Installation instructions are provided on the VSN International website https://www.vsni.co.uk.

1.2.2 Help and references

Documentation for the asreml() function, support functions and related methods are available in Windows help format, and in HTML form on Linux platforms. Typically, help is available via the

standard help mechanism; that is, help(asreml) or ?asreml displays the asreml documentation in text or HTML form depending on implementation and help system state.

There is an ASReml forum that all users are encouraged to join; visit
http://www.vsni.co.uk/forum to register.

The statistical theory underlying the modelling illustrated in this manual is introduced in Chapter 2. An extended discussion, with special reference to the fitting of variance models to structures at the residual ($\boldsymbol{R}$) and non-residual (random, $\boldsymbol{G}$) levels, will appear in detail in a forthcoming publication.

1.2.3 Using this guide

Users may find the introductory sections of Chapter 3 useful before reading further. This gives an introduction to analysis in ASReml-R using an example from the literature and covers some common tasks from creating a data frame to setting initial values for variance components. An introduction to the theory that underpins the methods in ASReml-R is covered in Chapter 2.

Variance modelling is a complex aspect of linear mixed modelling. Chapter 4 gives details of variance modelling in ASReml-R. You should refer to this chapter if you wish to fit more complex variance models. Chapter 5 describes the inclusion of known variance structures, such as those from ancestral pedigree information, in the model fitting process.

Prediction from the fitted linear mixed model is discussed in Chapter 6.

Chapter 7 presents a wide range of additional worked examples.

A complete description of the components of an asreml object and the data sets used in this manual are given in the ASReml-R Package Reference which is a pdf version of the ASReml help pages obtained in R by typing `help(asreml)`. This document is available at http://asreml.org under `Resources` > `ASReml docs` and on the VSN International website https://www.vsni.co.uk. Once ASReml-R is loaded, the data sets can be accessed using the data() function in R, for example:

```
> data(grass)
```

to access the `grass` data set.

1.3 Data sets used

1.3.1 Nebraska Intrastate Nursery (NIN) field experiment

The yield data from an advanced Nebraska Intrastate Nursery (NIN) breeding trial conducted at Alliance in 1988/89 are taken from Stroup et al. (1994). 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. In field trials, complete replicates are typically allocated to consecutive groups of *whole* columns or rows. In this trial the replicates were not allocated to groups of whole columns,

but rather, overlapped columns. Table 1.1 gives the allocation of varieties to plots in field plan order with replicates 1 and 3 in *italics* and replicates 2 and 4 in **bold**.

Table 1.1: Trial layout and allocation of varieties to plots in the NIN field trial

	column										
row	1	2	3	4	5	6	7	8	9	10	11
1	-	*NE83407*	*BUCKSKIN*	*NE87612*	**VONA**	**NE87512**	*NE87408*	*CODY*	*BUCKSKIN*	**NE87612**	**KS831374**
2	-	*CENTURA*	*NE86527*	*NE87613*	**NE87463**	**NE83407**	*NE83407*	*NE87612*	*NE83406*	**BUCKSKIN**	**NE86482**
3	-	*SCOUT66*	*NE86582*	*NE87615*	**NE86507**	**NE87403**	*NORKAN*	*NE87457*	*NE87409*	**NE85556**	**NE85623**
4	-	*COLT*	*NE86606*	*NE87619*	**BUCKSKIN**	**NE87457**	*REDLAND*	*NE84557*	*NE87499*	**BRULE**	**NE86527**
5	-	*NE83498*	*NE86607*	*NE87627*	**ROUGHRIDER**	**NE83406**	*KS831374*	*NE83T12*	*CENTURA*	**NE86507**	**NE87451**
6	-	*NE84557*	*ROUGHRIDER*	-	**NE86527**	**COLT**	*COLT*	*NE86507*	*NE83432*	**ROUGHRIDER**	**NE87409**
7	-	*NE83432*	*VONA*	**CENTURA**	**SCOUT66**	**NE87522**	*NE86527*	*TAM200*	*NE87512*	**VONA**	**GAGE**
8	-	*NE85556*	*SIOUXLAND*	**NE85623**	**NE86509**	**NORKAN**	*VONA*	*NE87613*	*ROUGHRIDER*	**NE83404**	**NE83407**
9	-	*NE85623*	*GAGE*	**CODY**	**NE86606**	**NE87615**	*TAM107*	*ARAPAHOE*	*NE83498*	**CODY**	**NE87615**
10	-	*CENTURAK78*	*NE83T12*	**NE86582**	**NE84557**	**NE85556**	*CENTURAK78*	*SCOUT66*	-	**NE87463**	**ARAPAHOE**
11	-	*NORKAN*	*NE86T666*	**NE87408**	**KS831374**	**TAM200**	*NE87627*	*NE87403*	**NE86T666**	**NE86582**	**CHEYENNE**
12	-	*KS831374*	*NE87403*	**NE87451**	**GAGE**	**LANCOTA**	*NE86T666*	*NE85623*	**NE87403**	**NE87499**	**REDLAND**
13	-	*TAM200*	*NE87408*	**NE83432**	**NE87619**	**NE86503**	*NE87615*	*NE86509*	**NE87512**	**NORKAN**	**NE83432**
14	-	*NE86482*	*NE87409*	**CENTURAK78**	**NE87499**	**NE86482**	*NE86501*	*NE85556*	**NE87446**	**SCOUT66**	**NE87619**
15	-	*HOMESTEAD*	*NE87446*	**NE83T12**	**CHEYENNE**	**BRULE**	*NE87522*	*HOMESTEAD*	**CENTURA**	**NE87513**	**NE83498**
16	*LANCER*	*LANCOTA*	*NE87451*	**NE87409**	**NE86607**	**NE87612**	*CHEYENNE*	*NE83404*	**NE86503**	**NE83T12**	**NE87613**
17	*BRULE*	*NE86501*	*NE87457*	**NE87513**	**NE83498**	**NE87613**	*SIOUXLAND*	*NE86503*	**NE87408**	**CENTURAK78**	**NE86501**
18	*REDLAND*	*NE86503*	*NE87463*	**NE87627**	**NE83404**	**NE86T666**	*NE87451*	*NE86582*	**COLT**	**NE87627**	**TAM200**
19	*CODY*	*NE86507*	*NE87499*	**ARAPAHOE**	**NE87446**	-	*GAGE*	*NE87619*	**LANCER**	**NE86606**	**NE87522**
20	*ARAPAHOE*	*NE86509*	*NE87512*	**LANCER**	**SIOUXLAND**	*NE86607*	*LANCER*	*NE87463*	**NE83406**	**NE87457**	**NE84557**
21	*NE83404*	*TAM107*	*NE87513*	**TAM107**	**HOMESTEAD**	*LANCOTA*	*NE87446*	*NE86606*	**NE86607**	**NE86509**	**TAM107**
22	*NE83406*	*CHEYENNE*	*NE87522*	**REDLAND**	**NE86501**	*NE87513*	*NE86482*	*BRULE*	**SIOUXLAND**	**LANCOTA**	**HOMESTEAD**

1.3.2 Repeated measures on rats

Growth curve data on the body weights of rats are taken from Box (1950). 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 and the raw results are shown in Figure 1.1.

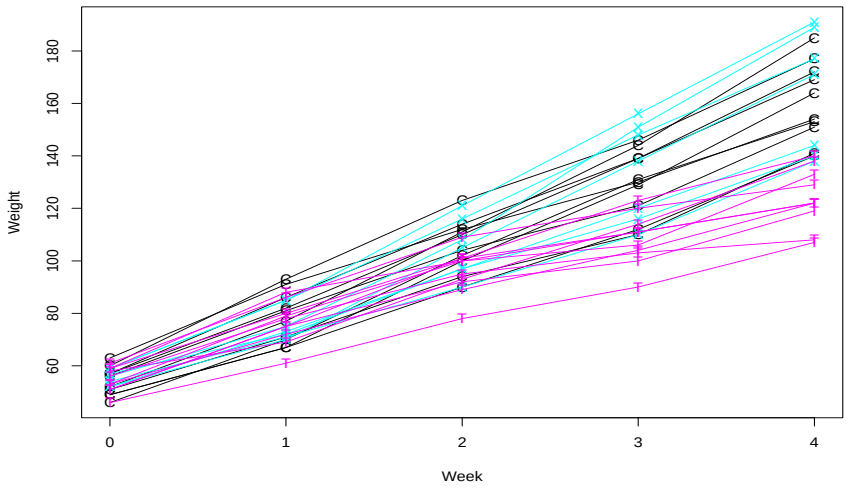

Figure 1.1: Weekly body weights of rats. C = Control, X = Thyroxin, T = Thiouracil

1.3.3 Orange wether trial

Three key traits for the Australian wool industry are the weight of wool grown per year, the cleanness and the diameter of that wool. Much of the wool is produced from wethers and most major producers have traditionally used a particular strain or bloodline. To assess the importance of bloodline differences, many wether trials were conducted. One trial was conducted from 1984 to 1988 at Borenore near Orange. It involved 35 teams of wethers representing 27 bloodlines. The following extract from the file `orange.csv` contains greasy fleece weight (kg), yield (percentage of clean fleece weight to greasy fleece weight) and fibre diameter (microns).

```
Tag, Site, Bloodline, Team, Year, gfw, yield, fdiam
0101, 3, 21, 1, 1, 5.6, 74.3, 18.5
0101, 3, 21, 1, 2, 6.0, 71.2, 19.6
0101, 3, 21, 1, 3, 8.0, 75.7, 21.5
0102, 3, 21, 1, 1, 5.3, 70.9, 20.8
0102, 3, 21, 1, 2, 5.7, 66.1, 20.9
0102, 3, 21, 1, 3, 6.8, 70.3, 22.1
0103, 3, 21, 1, 1, 5.0, 80.7, 18.9
0103, 3, 21, 1, 2, 5.5, 75.5, 19.9
0103, 3, 21, 1, 3, 7.0, 76.6, 21.9
⋮
4013, 3, 43, 35, 1, 7.9, 75.9, 22.6
```

```
4013, 3, 43, 35, 2, 7.8, 70.3, 23.9
4013, 3, 43, 35, 3, 9.0, 76.2, 25.4
4014, 3, 43, 35, 1, 8.3, 66.5, 22.2
4014, 3, 43, 35, 2, 7.8, 63.9, 23.3
4014, 3, 43, 35, 3, 9.9, 69.8, 25.5
4015, 3, 43, 35, 1, 6.9, 75.1, 20.0
4015, 3, 43, 35, 2, 7.6, 71.2, 20.3
4015, 3, 43, 35, 3, 8.5, 78.1, 21.7
```

1.3.4 Beef cattle data

These data appear among the examples in Harvey (1977) and are originally from Harvey (1960). The data comprise 65 observations on individual calves indexed by factors *Line* and *Sire* within *line*. The data as used here contain a covariate *ageOfDam* and 3 response variates *average daily gain*, *age* and *initial weight* labelled as *y1, y2* and *y3*, respectively.

An extract from `harvey.dat` is given below:

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
105	Sire_1	0	1	5	186	483	258
106	Sire_1	0	1	5	177	469	267
107	Sire_1	0	1	5	177	428	271
108	Sire_1	0	1	5	163	439	247
109	Sire_2	0	1	4	188	439	229
110	Sire_2	0	1	4	178	407	226
⋮							
161	Sire_9	0	3	4	184	483	244
162	Sire_9	0	3	5	180	425	266
163	Sire_9	0	3	5	177	420	246
164	Sire_9	0	3	5	175	449	252
165	Sire_9	0	3	5	164	405	242

In a genetic analysis we can specify the relationship among individuals in a pedigree file. This is a simple text file with columns for the individual's identity and its male and female parents. The first 20 lines of the pedigree file `harvey.ped` associated with these data are:

Calf	Sire	Dam
101	Sire_1	0
102	Sire_1	0
103	Sire_1	0
104	Sire_1	0
105	Sire_1	0

```
106  Sire_1  0
107  Sire_1  0
108  Sire_1  0
109  Sire_2  0
110  Sire_2  0
111  Sire_2  0
112  Sire_2  0
113  Sire_2  0
114  Sire_2  0
115  Sire_2  0
116  Sire_2  0
117  Sire_3  0
118  Sire_3  0
119  Sire_3  0
120  Sire_3  0
```

where unknown parents are denoted here by 0. In this example the columns of the pedigree file `harvey.ped` are fully contained within the data file `harvey.dat`.

2 Some theory

2.1 The general linear mixed model

If $\boldsymbol{y}$ $(n \times 1)$ denotes the vector of observations, the general linear mixed model can be written as

$$\boldsymbol{y} = \boldsymbol{X}\boldsymbol{\tau} + \boldsymbol{Z}\boldsymbol{u} + \boldsymbol{e} \tag{2.1}$$

where $\boldsymbol{\tau}$ $(p \times 1)$ is a vector of fixed effects, $\boldsymbol{X}$ $(n \times p)$ is the design matrix of full column rank that associates observations with the appropriate combination of fixed effects, $\boldsymbol{u}$ $(q \times 1)$ is a vector of random effects, $\boldsymbol{Z}$ $(n \times q)$ is the design matrix that associates observations with the appropriate combination of random effects, and $\boldsymbol{e}$ $(n \times 1)$ is the vector of residual errors.

2.1.1 Sigma parameterization of the linear mixed model

Model (2.1) is called a linear mixed model or linear mixed effects model. It is assumed

$$\begin{bmatrix} \boldsymbol{u} \\ \boldsymbol{e} \end{bmatrix} \sim N\left(\begin{bmatrix} \boldsymbol{0} \\ \boldsymbol{0} \end{bmatrix}, \begin{bmatrix} \boldsymbol{G}(\boldsymbol{\sigma}_g) & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{R}_v(\boldsymbol{\sigma}_r) \end{bmatrix} \right) \tag{2.2}$$

where the matrices $\boldsymbol{G}$ and $\boldsymbol{R}_v$ are variance matrices for $\boldsymbol{u}$ and $\boldsymbol{e}$ and are functions of parameters $\boldsymbol{\sigma}_g$ and $\boldsymbol{\sigma}_r$. This requires that the random effects $\boldsymbol{u}$ and residual errors $\boldsymbol{e}$ are uncorrelated. The variance matrix for $\boldsymbol{y}$ is then of the form

$$\text{var}\,(\boldsymbol{y}) \quad = \quad \boldsymbol{Z}\boldsymbol{G}(\boldsymbol{\sigma}_g)\boldsymbol{Z}^{\mathsf{T}} + \boldsymbol{R}_v(\boldsymbol{\sigma}_r) \tag{2.3}$$

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*. The variance models given by $\boldsymbol{G}$ and $\boldsymbol{R}_v$ are referred to as *G structures* and *R structures* respectively.

We illustrate these concepts using the simplest linear mixed model, that is, the one-way classification.

Example 2.1 A simple example Consider a one-way classification comprising a single random effect $\boldsymbol{u}$, and a residual error term $\boldsymbol{e}$. The two random components of this model, namely $\boldsymbol{u}$ and $\boldsymbol{e}$, are each assumed to be independent and identically distributed (IID) and to follow a normal distribution such that $\boldsymbol{u} \sim N(\boldsymbol{0}, \sigma_u^2 \boldsymbol{I}_q)$ and $\boldsymbol{e} \sim N(\boldsymbol{0}, \sigma_e^2 \boldsymbol{I}_n)$. Hence the variance of $\boldsymbol{y}$ has the form

$$\text{var}\,(\boldsymbol{y}) \quad = \quad \sigma_u^2 \boldsymbol{Z}\boldsymbol{Z}^{\mathsf{T}} + \sigma_e^2 \boldsymbol{I}_n \tag{2.4}$$

This model has two variance structure parameters or sigmas: the variance component σ_u^2 associated with $\boldsymbol{u}$, and the variance component σ_e^2 associated with $\boldsymbol{e}$. Mapping this equation back to (2.3), we have $\boldsymbol{\sigma}_g = \sigma_u^2$, $\boldsymbol{G}(\boldsymbol{\sigma}_g) = \sigma_u^2 \boldsymbol{I}_q$, $\boldsymbol{\sigma}_r = \sigma_e^2$ and $\boldsymbol{R}_v(\boldsymbol{\sigma}_r) = \sigma_e^2 \boldsymbol{I}_n$.

2.1.2 Partitioning the fixed and random model terms

Typically, $\boldsymbol{\tau}$ and $\boldsymbol{u}$ are composed of several model terms, that is, $\boldsymbol{\tau}$ can be partitioned as $\boldsymbol{\tau} = [\boldsymbol{\tau}_1^\intercal \dots \boldsymbol{\tau}_t^\intercal]^\intercal$ and $\boldsymbol{u}$ can be partitioned as $\boldsymbol{u} = [\boldsymbol{u}_1^\intercal \dots \boldsymbol{u}_b^\intercal]^\intercal$, with $\boldsymbol{X}$ and $\boldsymbol{Z}$ partitioned conformably as $\boldsymbol{X} = [\boldsymbol{X}_1 \dots \boldsymbol{X}_t]$ and $\boldsymbol{Z} = [\boldsymbol{Z}_1 \dots \boldsymbol{Z}_b]$.

2.1.3 G structure for the random model terms

For $\boldsymbol{u}$ partitioned as $\boldsymbol{u} = [\boldsymbol{u}_1^\intercal \dots \boldsymbol{u}_b^\intercal]^\intercal$, we impose a direct sum structure on the matrix $\boldsymbol{G}$, written

$$\boldsymbol{G} = \oplus_{i=1}^{b'} \boldsymbol{G}_i = \begin{bmatrix} \boldsymbol{G}_1 & 0 & \dots & 0 & 0 \\ 0 & \boldsymbol{G}_2 & \dots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & \boldsymbol{G}_{b'-1} & 0 \\ 0 & 0 & \dots & 0 & \boldsymbol{G}_{b'} \end{bmatrix}$$

where $\oplus$ is the direct sum operator, each $\boldsymbol{G}_i$ is of size q_i and $q = \sum_i q_i$.

The default assumption is that each random model term generates one component of this direct sum (then $b' = b$ and $\text{var}(\boldsymbol{u}_i) = \boldsymbol{G}_i$ for $i = 1 \dots b$). This means that the random effects from any two distinct model terms are uncorrelated. However, in some models, one component of $\boldsymbol{G}$ may apply across several model terms, for example, in random coefficient regression where the random intercepts and slopes for subjects are correlated. To accommodate these cases, one component of $\boldsymbol{G}$ may apply across several model terms (then $b' < b$).

Example 2.2 Variance components mixed models

Building example 2.1 to a linear mixed model with more than one ($b > 1$) random effect (typically known as a variance components mixed model), the random effects $\boldsymbol{u}_i$ in $\boldsymbol{u}$, and the residual errors $\boldsymbol{e}$, are assumed pairwise uncorrelated and to each be normally distributed with mean zero and variance given by

$$\text{var}(\boldsymbol{u}_i) = \sigma_{u_i}^2 \boldsymbol{I}_{q_i}$$

and

$$\text{var}(\boldsymbol{e}) = \sigma_e^2 \boldsymbol{I}_n$$

where $\boldsymbol{I}_{q_i}$ and $\boldsymbol{I}_n$ are identity matrices of dimension q_i and n, respectively. In this case

$$\text{var}(\boldsymbol{y}) = \sum_{i=1}^{b} \sigma_{u_i}^2 \boldsymbol{Z}_i \boldsymbol{Z}_i^\intercal + \sigma_e^2 \boldsymbol{I}_n. \tag{2.5}$$

2.1.4 Partitioning the residual error term

As for the fixed and random model terms, it is often useful or appropriate to consider a partitioning of the vector of residual errors $\boldsymbol{e}$ according to some conditioning factor. We use the term *section*

to describe this partitioning and the most common example of the use of sections in $\boldsymbol{e}$ is when we wish to allow sections in the data to have different variance structures. For example, in the analysis of multi-environment trials (METs) it is natural to expect that each trial will require a separate (possibly spatial) error structure. In this case, for s sections we have $\boldsymbol{e} = [\boldsymbol{e}_1^\intercal, \boldsymbol{e}_2^\intercal, \ldots, \boldsymbol{e}_s^\intercal]^\intercal$ assuming that the data vector is ordered by section, and where $\boldsymbol{e}_j$ represents the vector of errors for the j^{th} section.

2.1.5 R structure for the residual error term

For $\boldsymbol{e}$ partitioned as $\boldsymbol{e} = [\boldsymbol{e}_1^\intercal, \boldsymbol{e}_2^\intercal, \ldots, \boldsymbol{e}_s^\intercal]^\intercal$ we allow the matrix $\boldsymbol{R}_v$ to have a similar direct sum structure, with

$$\boldsymbol{R}_v = \oplus_{j=1}^{s} \boldsymbol{R}_{v_j} = \begin{bmatrix} \boldsymbol{R}_{v_1} & 0 & \ldots & 0 & 0 \\ 0 & \boldsymbol{R}_{v_2} & \ldots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \ldots & \boldsymbol{R}_{v_{s-1}} & 0 \\ 0 & 0 & \ldots & 0 & \boldsymbol{R}_{v_s} \end{bmatrix}$$

for $s \geq 1$ sections and the data ordered by section. Note that it may be necessary to re-order (re-number) the data units in order to achieve this structure. In ASReml-R it is now straightforward to apply possibly different variance structures to each component of $\boldsymbol{R}_v$.

In many cases, the residual errors ($\boldsymbol{e}$) can be expected to share a common variance structure. In this case there is only one section ($s = 1$).

Typically a variance structure is specified for each random model term and often more complex models than the simple IID model are specified. ASReml-R offers a wide range of variance models to choose from. A full listing is in Table B.1 and details are provided in Chapter 4.

2.1.6 Gamma parameterization for the linear mixed model

The sigma parameterization of model (2.3) is one possible parameterization of $\text{var}(\boldsymbol{y})$. In this parameterization both $\boldsymbol{G}(\boldsymbol{\sigma}_g)$ and $\boldsymbol{R}_v(\boldsymbol{\sigma}_r)$ are variance matrices and the variance structure parameters in $\boldsymbol{\sigma}_g$ and $\boldsymbol{\sigma}_r$ are referred to as *sigmas,* see above. Other parameterizations are possible and are sometimes useful. For example, in some of the early development of REML for the traditional mixed model of (2.5), the variance matrix was parameterized as the equivalent model

$$\text{var}(\boldsymbol{y}) \quad = \quad \sigma_e^2 \left(\sum_i^b \gamma_{g_i} \boldsymbol{Z}_i \boldsymbol{Z}_i^\intercal + \boldsymbol{I}_n \right) \tag{2.6}$$

for γ_{g_i} being the ratio of the variance component for the random term $\boldsymbol{u}_i$ relative to error variance, that is, $\gamma_{g_i} = \sigma_{u_i}^2 / \sigma_e^2$. In this case ASReml-R calculated a simple estimate of σ_e^2 and initial values for the iterative process were specified in terms of the ratios γ_{g_i} rather than in terms of the variance components $\sigma_{u_i}^2$. It was often easier to specify initial values in terms of these ratios rather than the variance components which is why this approach was adopted. Where $\boldsymbol{R}_v(\boldsymbol{\sigma}_r)$ can be written as a scaled correlation matrix, that is, $\boldsymbol{R}_v(\boldsymbol{\sigma}_r) = \sigma_e^2 \boldsymbol{R}_c(\boldsymbol{\gamma}_r)$, this suggests the alternative specification of (2.2)

$$\begin{bmatrix} \boldsymbol{u} \\ \boldsymbol{e} \end{bmatrix} \sim N \left(\begin{bmatrix} \boldsymbol{0} \\ \boldsymbol{0} \end{bmatrix}, \sigma_e^2 \begin{bmatrix} \boldsymbol{G}(\boldsymbol{\gamma}_g) & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{R}_c(\boldsymbol{\gamma}_r) \end{bmatrix} \right) \tag{2.7}$$

where $\boldsymbol{\gamma}_g$ and $\boldsymbol{\gamma}_r$ represent the variance structure parameters associated with scaled (by σ_e^2) variance matrices. In this case

$$\text{var}\,(\boldsymbol{y}) \quad = \quad \sigma_e^2 \left(\boldsymbol{Z}\boldsymbol{G}(\boldsymbol{\gamma}_g)\boldsymbol{Z}^{\mathsf{T}} + \boldsymbol{R}_c(\boldsymbol{\gamma}_r)\right), \tag{2.8}$$

which we will refer to as the *gamma parameterization*, and the individual variance structure parameters in $\boldsymbol{\gamma}_g$ and $\boldsymbol{\gamma}_r$ will be referred to as *gammas*. ASReml-R switches between the sigma and gamma parameterizations for estimation. This is discussed in Section 4.8.

2.1.7 Parameter types

Each sigma in $\boldsymbol{\sigma}_g$ and $\boldsymbol{\sigma}_r$ and each gamma in $\boldsymbol{\gamma}_g$ and $\boldsymbol{\gamma}_r$ has a parameter type, for example, variance components, variance component ratios, autocorrelation parameters, factor loadings. Furthermore, the parameters in $\boldsymbol{\sigma}_g$, $\boldsymbol{\sigma}_r$, $\boldsymbol{\gamma}_g$ and $\boldsymbol{\gamma}_r$ can span multiple types. For example, the spatial analysis of a simple column trial would involve variance components (sigma parameterization) or variance component ratios (gamma parameterization) and spatial autocorrelation parameters.

2.1.8 Variance structures for the random model terms

The random model terms $\boldsymbol{u}_i$ in $\boldsymbol{u}$ define the random effects and associated design matrices, $\boldsymbol{Z}_i \in \boldsymbol{Z}$, but additional information is required before the model can be fitted. This extra step involves defining the G structure for each term. In Release 4, this is achieved by using functions to directly apply variance models to the individual component factors in a random model term to define $\boldsymbol{G}_i$. This produces a consolidated model term that simultaneously defines both the design matrix ($\boldsymbol{Z}_i$) and variance model ($\boldsymbol{G}_i$). This process is described in detail in Chapter 4 with examples.

2.1.9 Variance models for terms with several factors

A random model term may comprise either a single factor or several component factors to give a compound model term. Consider a compound model term represented by `A:B`, where the component factors `A` and `B` have a and b levels, respectively. The vector $\boldsymbol{u}_{AB}$ for `A:B` is generated with the levels of `B` nested in the levels of `A`, that is, the levels of `B` cycling fastest:

$$\boldsymbol{u}_{AB} \quad = \quad (u_{11},\ u_{12}, \ldots u_{1b}, u_{21},\ u_{22}, \ldots u_{2b}, \ldots, u_{a1},\ u_{a2}, \ldots u_{ab})^{\mathsf{T}}.$$

To illustrate the variance structure clearly, let $\boldsymbol{u}_{iB}$ be the vector with b elements containing effects for the ith level of `A` and $\boldsymbol{u}_{Ak}$ be the vector with a elements containing effects for the kth level of `B`. Now consider the variance model for the term `A:B`. Let $\boldsymbol{\Sigma}_A = [a_{ij}]$ be an $a \times a$ symmetric variance matrix and let $\boldsymbol{\Sigma}_B = [b_{ij}]$ be a $b \times b$ symmetric variance matrix. The assumption of separability (for example, Gelfand et al. (2010)) suggests modelling the variances using $cov\left(\boldsymbol{u}_{iB}, \boldsymbol{u}_{jB}^{\mathsf{T}}\right)$ as $a_{ij}\boldsymbol{\Sigma}_B$. This model is called *separable* as we have factorized the covariances into terms dependent on factor `A` (a_{ij}) and on factor `B` ($\boldsymbol{\Sigma}_B$). We find

$$\boldsymbol{\Sigma}_{u_{AB}} \quad = \quad \text{var}\,(\boldsymbol{u}_{AB}) \quad = \quad \boldsymbol{\Sigma}_A \otimes \boldsymbol{\Sigma}_B,$$

see Section 2.1.10 for the mathematical definition of a direct product structure, and

$$cov\,(u_{ik}, u_{jl}) \quad = \quad a_{ij} \times b_{kl}.$$

The latter is easily obtained by constructing $cov\left(\boldsymbol{u}_{iB}, \boldsymbol{u}_{jB}^{\intercal}\right) = a_{ij}\boldsymbol{\Sigma}_B$ and then extracting the (k,l)th term $a_{ij}b_{kl}$. Two simple examples are:

- $\boldsymbol{\Sigma}_{u_{AB}} = \boldsymbol{I}_A \otimes \boldsymbol{\Sigma}_B$ where $\boldsymbol{\Sigma}_B$ has an unstructured form. This means that the elements of $\boldsymbol{u}_{iB}$ are IID $N(\mathbf{0},\ \boldsymbol{\Sigma}_B)$. This model is widely used in the analysis of multivariate data.

- $\boldsymbol{\Sigma}_{u_{AB}} = \boldsymbol{\Sigma}_A \otimes \boldsymbol{I}_B$ where $\boldsymbol{\Sigma}_A$ represents a first order autoregressive process. This means that the elements of $\boldsymbol{u}_{Ak}$ are IID realisations of this process. This model is used widely in the analysis of field trial data to model spatial trend in one direction only.

Example 2.3 A simple separable structure

If `A` has 3 levels and `B` has 2 levels, then the vector $\boldsymbol{u}_{AB}$ for the term `A:B` would be

$$\boldsymbol{u}_{AB} = (u_{11},\ u_{12},\ u_{21},\ u_{22},\ u_{31},\ u_{32})^{\intercal}.$$

Let $\boldsymbol{\Sigma}_{u_{AB}} = \boldsymbol{\Sigma}_A \otimes \boldsymbol{\Sigma}_B$ where $\boldsymbol{\Sigma}_A = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}$ and $\boldsymbol{\Sigma}_B = \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix}$.

If $\boldsymbol{u}_{A1}$ the set of effects in $\boldsymbol{u}_{AB}$ for level 1 of `B` and $\boldsymbol{u}_{A2}$ is the set of effects in $\boldsymbol{u}_{AB}$ for level 2 of `B`, then $cov\left(\boldsymbol{u}_{A1}, \boldsymbol{u}_{A1}^{\intercal}\right) = \text{var}\left(\boldsymbol{u}_{A1}\right) = b_{11}\boldsymbol{\Sigma}_A$ and $cov\left(\boldsymbol{u}_{A1}, \boldsymbol{u}_{A2}^{\intercal}\right) = b_{12}\boldsymbol{\Sigma}_A$. Similarly for $\boldsymbol{u}_{1B}$, $\boldsymbol{u}_{2B}$ and $\boldsymbol{u}_{3B}$ being the combined vectors of effects for each level of `A`. As examples, $cov\left(\boldsymbol{u}_{1B}, \boldsymbol{u}_{1B}^{\intercal}\right) = \text{var}\left(\boldsymbol{u}_{1B}\right) = a_{11}\boldsymbol{\Sigma}_B$ and $cov\left(\boldsymbol{u}_{2B}, \boldsymbol{u}_{3B}^{\intercal}\right) = a_{23}\boldsymbol{\Sigma}_B$. Finally, using magenta and blue to highlight terms associated with `A` and `B`, respectively,

$$cov\left(u_{21}, u_{32}\right) = a_{23} \times b_{12}.$$

2.1.10 Direct product structures

For the matrix $\boldsymbol{A} = \begin{bmatrix} a_{11} & \dots & a_{1p} \\ \vdots & \ddots & \vdots \\ a_{m1} & \dots & a_{mp} \end{bmatrix}$ and a second matrix $\boldsymbol{B}$, the *direct product structure* of $\boldsymbol{A}$ and $\boldsymbol{B}$ is written in full as

$$\boldsymbol{A} \otimes \boldsymbol{B} = \begin{bmatrix} a_{11}\boldsymbol{B} & \dots & a_{1p}\boldsymbol{B} \\ \vdots & \ddots & \vdots \\ a_{m1}\boldsymbol{B} & \dots & a_{mp}\boldsymbol{B} \end{bmatrix}.$$

As explained in the previous section, structures associated with direct product construction are known as *separable* variance structures and we call the assumption that a separable variance structure is plausible the *assumption of separability.*

2.1.11 Direct products in R structures

Separable structures occur naturally in many practical situations. Consider a vector of common errors associated with an experiment. The usual least squares assumption (and the default in

ASReml-R) is that these are independently and identically distributed (IID). However, if $\boldsymbol{e}$ was from a field experiment laid out in a rectangular array of r rows by c columns, we could arrange the residuals as a matrix and might consider that they were autocorrelated within rows and columns. Writing the residuals as a vector in field order, that is, by sorting the residuals rows within columns (plots within blocks) the variance of the residuals might then be

$$\sigma_e^2 \, \boldsymbol{\Sigma}_c(\rho_c) \otimes \boldsymbol{\Sigma}_r(\rho_r)$$

where $\boldsymbol{\Sigma}_c(\rho_c)$ and $\boldsymbol{\Sigma}_r(\rho_r)$ are correlation matrices for the row model (order r, autocorrelation parameter ρ_r) and column model (order c, autocorrelation parameter ρ_c) respectively. More specifically, a two-dimensional separable autoregressive spatial structure (AR1 $\otimes$ AR1) is sometimes assumed for the common errors in a field trial analysis (see Gogel (1997) and Cullis *et al.* (1998) for examples). In this case

$$\boldsymbol{\Sigma}_r = \begin{bmatrix} 1 & & & & \\ \rho_r & 1 & & & \\ \rho_r^2 & \rho_r & 1 & & \\ \vdots & \vdots & \vdots & \ddots & \\ \rho_r^{r-1} & \rho_r^{r-2} & \rho_r^{r-3} & \dots & 1 \end{bmatrix} \quad \text{and} \quad \boldsymbol{\Sigma}_c = \begin{bmatrix} 1 & & & & \\ \rho_c & 1 & & & \\ \rho_c^2 & \rho_c & 1 & & \\ \vdots & \vdots & \vdots & \ddots & \\ \rho_c^{c-1} & \rho_c^{c-2} & \rho_c^{c-3} & \dots & 1 \end{bmatrix}.$$

Alternatively, the residuals might relate to a multivariate analysis with n_t traits and n units and be ordered traits *within* units. In this case an appropriate variance structure might be

$$\boldsymbol{I}_n \otimes \boldsymbol{\Sigma}$$

where $\boldsymbol{\Sigma}^{(n_t \times n_t)}$ is a general or *unstructured* variance matrix. See Chapter 4 for details on specifying separable R structures in ASReml-R.

2.1.12 Direct products in G structures

Likewise, the random model terms in $\boldsymbol{u}$ may have a direct product variance structure. For example, for a field trial with s sites, g varieties and the effects ordered varieties *within* sites, the random model term *site.variety* may have the variance structure

$$\boldsymbol{\Sigma} \otimes \boldsymbol{I}_g$$

where $\boldsymbol{\Sigma}$ is the variance matrix for sites. This would imply that the varieties are independent random effects within each site, have different variances at each site, and are correlated across sites. **Important** Whenever a random term is formed as the interaction of two factors you should consider whether the IID assumption is sufficient or if a direct product structure might be more appropriate. See Chapter 4 for details on specifying separable G structures in ASReml-R.

2.1.13 Range of variance models for R and G structures

A range of models are available for the components of both R and G structures. They include correlation (C) models (that is, where the diagonals are 1), or covariance (V) models and are discussed in detail in Chapter 4. Among the range of correlation models are:

- identity (that is, independent and identically distributed with variance 1)

- autoregressive (order 1 or 2)
- moving average (order 1 or 2)
- ARMA(1,1)
- uniform
- banded
- general correlation.

Among the range of covariance models are:

- scaled identity (that is, independent and identically distributed with homogeneous variances)
- diagonal (that is, independent with heterogeneous variances)
- antedependence
- unstructured
- factor analytic
- reduced rank.

There is also the facility to define models based on relationship matrices, including additive relationship matrices generated by pedigrees and using user specified variance matrices.

2.1.14 Combining variance models in R and G structures

The combination of variance models in separable G and R structures is a difficult and important concept. This is discussed in detail in Chapter 4.

2.2 Estimation

Consider the sigma parameterization of Section 2.1.1. Estimation involves two processes that are closely linked. They are performed within the 'engine' of ASReml-R. One process involves estimation of $\boldsymbol{\tau}$ and prediction of $\boldsymbol{u}$ (although the latter may not always be of interest) for given $\boldsymbol{\sigma}_g$ and $\boldsymbol{\sigma}_r$. The other process involves estimation of these variance parameters.

2.2.1 Estimation of the variance parameters

Estimation of the variance parameters is carried out using residual or restricted maximum likelihood (REML), developed by Patterson and Thompson (1971). An historical development of the theory

can be found in Searle *et al.* (1992). Note firstly that

$$\boldsymbol{y} \sim N(\boldsymbol{X}\boldsymbol{\tau},\ \boldsymbol{H}), \tag{2.9}$$

where $\boldsymbol{H} = \boldsymbol{Z}\boldsymbol{G}(\boldsymbol{\sigma}_g)\boldsymbol{Z}^\intercal + \boldsymbol{R}_v(\boldsymbol{\sigma}_r)$. REML does not use (2.9) for estimation of variance parameters, but rather uses a distribution free of $\boldsymbol{\tau}$, essentially based on error contrasts or *residuals*. The derivation given below is presented in Verbyla (1990).

We transform $\boldsymbol{y}$ using a non-singular matrix $\boldsymbol{L} = [\boldsymbol{L}_1\ \boldsymbol{L}_2]$ such that

$$\boldsymbol{L}_1^\intercal \boldsymbol{X} = I_p, \quad \boldsymbol{L}_2^\intercal \boldsymbol{X} = \boldsymbol{0}.$$

If $\boldsymbol{y}_j = \boldsymbol{L}_j^\intercal \boldsymbol{y}$, $j = 1, 2$,

$$\begin{bmatrix} \boldsymbol{y}_1 \\ \boldsymbol{y}_2 \end{bmatrix} \sim N\left(\begin{bmatrix} \boldsymbol{\tau} \\ \boldsymbol{0} \end{bmatrix}, \begin{bmatrix} \boldsymbol{L}_1^\intercal \boldsymbol{H} \boldsymbol{L}_1 & \boldsymbol{L}_1^\intercal \boldsymbol{H} \boldsymbol{L}_2 \\ \boldsymbol{L}_2^\intercal \boldsymbol{H} \boldsymbol{L}_1 & \boldsymbol{L}_2^\intercal \boldsymbol{H} \boldsymbol{L}_2 \end{bmatrix} \right).$$

The full distribution of $\boldsymbol{L}^\intercal \boldsymbol{y}$ can be partitioned into a *conditional distribution*, namely $\boldsymbol{y}_1 | \boldsymbol{y}_2$, for estimation of $\boldsymbol{\tau}$, and a *marginal distribution* based on $\boldsymbol{y}_2$ for estimation of $\boldsymbol{\sigma}_g$ and $\boldsymbol{\sigma}_r$; the latter is the basis of the **r**esidual likelihood.

The estimate of $\boldsymbol{\tau}$ is found by equating $\boldsymbol{y}_1$ to its conditional expectation, and after some algebra we find,

$$\hat{\boldsymbol{\tau}} = (\boldsymbol{X}^\intercal \boldsymbol{H}^{-1} \boldsymbol{X})^{-1} \boldsymbol{X}^\intercal \boldsymbol{H}^{-1} \boldsymbol{y}$$

Estimation of $\boldsymbol{\kappa} = [\boldsymbol{\sigma}_g^\intercal\ \boldsymbol{\sigma}_r^\intercal]^\intercal$ is based on the log residual likelihood,

$$\begin{aligned} \ell_R &= -\frac{1}{2}(\log\det \boldsymbol{L}_2^\intercal \boldsymbol{H}^{-1} \boldsymbol{L}_2 + \boldsymbol{y}_2^\intercal (\boldsymbol{L}_2^\intercal \boldsymbol{H} \boldsymbol{L}_2)^{-1} \boldsymbol{y}_2) \\ &= -\frac{1}{2}(\log\det \boldsymbol{X}^\intercal \boldsymbol{H}^{-1} \boldsymbol{X} + \log\det \boldsymbol{H} + \boldsymbol{y}^\intercal \boldsymbol{P} \boldsymbol{y}) \end{aligned} \tag{2.10}$$

where

$$\boldsymbol{P} = \boldsymbol{H}^{-1} - \boldsymbol{H}^{-1}\boldsymbol{X}(\boldsymbol{X}^\intercal \boldsymbol{H}^{-1} \boldsymbol{X})^{-1} \boldsymbol{X}^\intercal \boldsymbol{H}^{-1}.$$

Note that $\boldsymbol{y}^\intercal \boldsymbol{P} \boldsymbol{y} = (\boldsymbol{y} - \boldsymbol{X}\hat{\boldsymbol{\tau}})^\intercal \boldsymbol{H}^{-1} (\boldsymbol{y} - \boldsymbol{X}\hat{\boldsymbol{\tau}})$. The log-likelihood (2.10) depends on $\boldsymbol{X}$ and not on the particular non-unique transformation defined by $\boldsymbol{L}$.

The log residual likelihood (ignoring constants) can be written as

$$\ell_R = -\frac{1}{2}(\log\det \boldsymbol{C} + \log\det \boldsymbol{R}_v + \log\det \boldsymbol{G} + \boldsymbol{y}^\intercal \boldsymbol{P} \boldsymbol{y}). \tag{2.11}$$

We can also write

$$\boldsymbol{P} = \boldsymbol{R}_v^{-1} - \boldsymbol{R}_v^{-1} \boldsymbol{W} \boldsymbol{C}^{-1} \boldsymbol{W}^\intercal \boldsymbol{R}_v^{-1}$$

with $\boldsymbol{W} = [\boldsymbol{X}\ \ \boldsymbol{Z}]$. Letting $\boldsymbol{\kappa} = [\boldsymbol{\sigma}_g^\intercal\ \boldsymbol{\sigma}_r^\intercal]^\intercal$, the REML estimates of κ_i are found by calculating the score

$$U(\kappa_i) = \partial \ell_R / \partial \kappa_i = -\frac{1}{2}[\mathrm{tr}\,(\boldsymbol{P}\boldsymbol{H}_i) - \boldsymbol{y}^\intercal \boldsymbol{P} \boldsymbol{H}_i \boldsymbol{P} \boldsymbol{y}] \tag{2.12}$$

and equating to zero. Note that $\boldsymbol{H}_i = \partial \boldsymbol{H} / \partial \kappa_i$.

The elements of the observed information matrix are

$$-\frac{\partial^2 \ell_R}{\partial \kappa_i \partial \kappa_j} = \frac{1}{2}\mathrm{tr}\left(\boldsymbol{P}\boldsymbol{H}_{ij}\right) - \frac{1}{2}tr\left(\boldsymbol{P}\boldsymbol{H}_i\boldsymbol{P}\boldsymbol{H}_j\right) + \boldsymbol{y}^{\mathsf{T}}\boldsymbol{P}\boldsymbol{H}_i\boldsymbol{P}\boldsymbol{H}_j\boldsymbol{P}\boldsymbol{y} - \frac{1}{2}\boldsymbol{y}^{\mathsf{T}}\boldsymbol{P}\boldsymbol{H}_{ij}\boldsymbol{P}\boldsymbol{y} \tag{2.13}$$

where $\boldsymbol{H}_{ij} = \partial^2 \boldsymbol{H}/\partial\kappa_i\partial\kappa_j$.

The elements of the expected information matrix are

$$\mathrm{E}\left(-\frac{\partial^2 \ell_R}{\partial \kappa_i \partial \kappa_j}\right) = \frac{1}{2}tr\left(\boldsymbol{P}\boldsymbol{H}_i\boldsymbol{P}\boldsymbol{H}_j\right). \tag{2.14}$$

Given an initial estimate $\boldsymbol{\kappa}^{(0)}$, an update of $\boldsymbol{\kappa}$, $\boldsymbol{\kappa}^{(1)}$ using the Fisher-scoring (FS) algorithm is

$$\boldsymbol{\kappa}^{(1)} = \boldsymbol{\kappa}^{(0)} + \boldsymbol{I}(\boldsymbol{\kappa}^{(0)}, \boldsymbol{\kappa}^{(0)})^{-1}\boldsymbol{U}(\boldsymbol{\kappa}^{(0)}) \tag{2.15}$$

where $\boldsymbol{U}(\boldsymbol{\kappa}^{(0)})$ is the score vector (2.12) and $\boldsymbol{I}(\boldsymbol{\kappa}^{(0)}, \boldsymbol{\kappa}^{(0)})$ is the expected information matrix (2.14) of $\boldsymbol{\kappa}$ evaluated at $\boldsymbol{\kappa}^{(0)}$.

For large models or large data sets, the evaluation of the trace terms in either (2.13) or (2.14) is either not feasible or is very computer intensive. To overcome this problem ASReml-R uses the AI algorithm (Gilmour, Thompson and Cullis, 1995). The matrix denoted by $\mathcal{I}_A$ is obtained by averaging (2.13) and (2.14) and approximating $\boldsymbol{y}^{\mathsf{T}}\boldsymbol{P}\boldsymbol{H}_{ij}\boldsymbol{P}\boldsymbol{y}$ by its expectation, $tr\left(\boldsymbol{P}\boldsymbol{H}_{ij}\right)$ in those cases when $\boldsymbol{H}_{ij} \neq 0$. For variance components models (that is those linear with respect to variances in $\boldsymbol{H}$), the terms in $\mathcal{I}_A$ are exact averages of those in (2.13) and (2.14). The basic idea is to use $\mathcal{I}_A(\kappa_i, \kappa_j)$ in place of the expected information matrix in (2.15) to update $\boldsymbol{\kappa}$.

The elements of $\mathcal{I}_A$ are

$$\mathcal{I}_{\mathcal{A}}(\kappa_i, \kappa_j) = \frac{1}{2}\boldsymbol{y}^{\mathsf{T}}\boldsymbol{P}\boldsymbol{H}_i\boldsymbol{P}\boldsymbol{H}_j\boldsymbol{P}\boldsymbol{y}. \tag{2.16}$$

The $\mathcal{I}_A$ matrix is the (scaled) residual sums of squares and products matrix of

$$\boldsymbol{y} = [\boldsymbol{y}_1, \ldots, \boldsymbol{y}_k]$$

where $\boldsymbol{y}_i$ is the 'working' variate for $\boldsymbol{\kappa}_i$ and is given by

$$\begin{aligned} \boldsymbol{y}_i &= \boldsymbol{H}_i\boldsymbol{P}\boldsymbol{y} \\ &= \boldsymbol{H}_i\boldsymbol{R}_v^{-1}\tilde{\boldsymbol{e}} \\ &= \boldsymbol{R}_{v_i}\boldsymbol{R}_v^{-1}\tilde{\boldsymbol{e}}, \quad \kappa_i \in \boldsymbol{\sigma}_r \\ &= \boldsymbol{Z}\boldsymbol{G}_i\boldsymbol{G}^{-1}\tilde{\boldsymbol{u}}, \quad \kappa_i \in \boldsymbol{\sigma}_g \end{aligned}$$

where $\tilde{\boldsymbol{e}} = \boldsymbol{y} - \boldsymbol{X}\hat{\boldsymbol{\tau}} - \boldsymbol{Z}\tilde{\boldsymbol{u}}$, $\hat{\boldsymbol{\tau}}$ and $\tilde{\boldsymbol{u}}$ are solutions to (2.17). In this form the AI matrix is relatively straightforward to calculate.

The combination of the AI algorithm with sparse matrix methods, in which only non-zero values are stored, gives an efficient algorithm in terms of both computing time and workspace.

2.2.2 Estimation/prediction of the fixed and random effects

To estimate $\boldsymbol{\tau}$ and predict $\boldsymbol{u}$ the objective function

$$\log f_{\boldsymbol{Y}}(\boldsymbol{y} \mid \boldsymbol{u}\ ;\ \boldsymbol{\tau}, \boldsymbol{R}_v) + \log f_{\boldsymbol{U}}(\boldsymbol{u}\ ;\ \boldsymbol{G})$$

is used. This is the log-joint distribution of $(\boldsymbol{Y}, \boldsymbol{u})$.

Differentiating with respect to $\boldsymbol{\tau}$ and $\boldsymbol{u}$ leads to the mixed model equations (Henderson *et al.*, 1959, Robinson, 1991) which are given by

$$\begin{bmatrix} \boldsymbol{X}^\mathsf{T}\boldsymbol{R}_v^{-1}\boldsymbol{X} & \boldsymbol{X}^\mathsf{T}\boldsymbol{R}_v^{-1}\boldsymbol{Z} \\ \boldsymbol{Z}^\mathsf{T}\boldsymbol{R}_v^{-1}\boldsymbol{X} & \boldsymbol{Z}^\mathsf{T}\boldsymbol{R}_v^{-1}\boldsymbol{Z} + \boldsymbol{G}^{-1} \end{bmatrix} \begin{bmatrix} \hat{\boldsymbol{\tau}} \\ \tilde{\boldsymbol{u}} \end{bmatrix} = \begin{bmatrix} \boldsymbol{X}^\mathsf{T}\boldsymbol{R}_v^{-1}\boldsymbol{y} \\ \boldsymbol{Z}^\mathsf{T}\boldsymbol{R}_v^{-1}\boldsymbol{y} \end{bmatrix}. \tag{2.17}$$

These can be written as

$$\boldsymbol{C}\tilde{\boldsymbol{\beta}} = \boldsymbol{W}^\mathsf{T}\boldsymbol{R}_v^{-1}\boldsymbol{y}$$

where $\boldsymbol{C} = \boldsymbol{W}^\mathsf{T}\boldsymbol{R}_v^{-1}\boldsymbol{W} + \boldsymbol{G}^*$, $\tilde{\boldsymbol{\beta}} = \begin{bmatrix}\hat{\boldsymbol{\tau}}^\mathsf{T} & \tilde{\boldsymbol{u}}^\mathsf{T}\end{bmatrix}^\mathsf{T}$ and

$$\boldsymbol{G}^* = \begin{bmatrix} \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{G}^{-1} \end{bmatrix}.$$

The solution of (2.17) requires values for $\boldsymbol{\sigma}_g$ and $\boldsymbol{\sigma}_r$. In practice we replace $\boldsymbol{\sigma}_g$ and $\boldsymbol{\sigma}_r$ by their REML estimates $\hat{\boldsymbol{\sigma}}_g$ and $\hat{\boldsymbol{\sigma}}_r$.

Note that $\hat{\boldsymbol{\tau}}$ is the empirical best linear unbiased estimator (E-BLUE) of $\boldsymbol{\tau}$, while $\tilde{\boldsymbol{u}}$ is the empirical best linear unbiased predictor (E-BLUP) of $\boldsymbol{u}$ for known $\boldsymbol{\sigma}_g$ and $\boldsymbol{\sigma}_r$. We also note that

$$\tilde{\boldsymbol{\beta}} - \boldsymbol{\beta} = \begin{bmatrix} \hat{\boldsymbol{\tau}} - \boldsymbol{\tau} \\ \tilde{\boldsymbol{u}} - \boldsymbol{u} \end{bmatrix} \sim N\left(\begin{bmatrix} \boldsymbol{0} \\ \boldsymbol{0} \end{bmatrix}, \boldsymbol{C}^{-1} \right).$$

2.2.3 Use of the gamma parameterization

ASReml-R uses either the gamma or sigma parameterization for estimation depending on the residual specification. The current default for univariate, single section data-sets is the gamma parameterization. In this case, all scale parameters are estimated as a ratio with respect to the residual variance, σ_e^2, and any parameters that measure only correlation are unchanged. See Chapter 4 for more detail.

2.3 What are BLUPs?

Consider a balanced one-way classification. For data records ordered r repeats within b treatments regarded as random effects, the linear mixed model is $\boldsymbol{y} = \boldsymbol{X}\boldsymbol{\tau} + \boldsymbol{Z}\boldsymbol{u} + \boldsymbol{e}$ where $\boldsymbol{X} = \boldsymbol{1}_b \otimes \boldsymbol{1}_r$ is the design matrix for $\boldsymbol{\tau}$ (the overall mean), $\boldsymbol{Z} = \boldsymbol{I}_b \otimes \boldsymbol{1}_r$ is the design matrix for the b (random) treatment effects u_i and $\boldsymbol{e}$ is the error vector. Assuming that the treatment effects are random implies that $\boldsymbol{u} \sim N(\boldsymbol{A}\boldsymbol{\psi},\ \sigma_b^2\boldsymbol{I}_b)$, for some design matrix $\boldsymbol{A}$ and parameter vector $\boldsymbol{\psi}$. It can be shown that

$$\tilde{\boldsymbol{u}} = \frac{r\sigma_b^2}{r\sigma_b^2 + \sigma^2}(\bar{\boldsymbol{y}} - \boldsymbol{1}\bar{y}_{\cdot\cdot}) + \frac{\sigma^2}{r\sigma_b^2 + \sigma^2}\boldsymbol{A}\boldsymbol{\psi} \tag{2.18}$$

where $\bar{\boldsymbol{y}}$ is the vector of treatment means, $\bar{y}_{\cdot\cdot}$ is the grand mean. The differences of the treatment means and the grand mean are the estimates of treatment effects if treatment effects are fixed. The

BLUP is therefore a weighted mean of the data based estimate and the 'prior' mean $\boldsymbol{A}\boldsymbol{\psi}$. If $\boldsymbol{\psi} = \mathbf{0}$, the BLUP in (2.18) becomes

$$\tilde{\boldsymbol{u}} = \frac{r\sigma_b^2}{r\sigma_b^2 + \sigma^2}(\bar{\boldsymbol{y}} - \mathbf{1}\bar{y}_{..}) \tag{2.19}$$

and the BLUP is a so-called shrinkage estimate. As $r\sigma_b^2$ becomes large relative to σ^2, the BLUP tends to the fixed effect solution, while for small $r\sigma_b^2$ relative to σ^2 the BLUP tends towards zero, the assumed initial mean. Thus (2.19) represents a weighted mean which involves the prior assumption that the u_i have zero mean.

Note also that the BLUPs in this simple case are constrained to sum to zero. This is essentially because the unit vector defining $\boldsymbol{X}$ can be found by summing the columns of the $\boldsymbol{Z}$ matrix. This linear dependence of the matrices translates to dependence of the BLUPs and hence constraints. This aspect occurs whenever the column space of $\boldsymbol{X}$ is contained in the column space of $\boldsymbol{Z}$. The dependence is slightly more complex with correlated random effects.

2.4 Inference: Random effects

2.4.1 Tests of hypotheses: variance parameters

Inference concerning variance parameters of a linear mixed effects model usually relies on approximate distributions for the (RE)ML estimates derived from asymptotic results.

It can be shown that the approximate variance matrix for the REML estimates is given by the inverse of the expected information matrix (Cox and Hinkley, 1974, section 4.8). Since this matrix is not available in ASReml-R we replace the expected information matrix by the AI matrix. Furthermore the REML estimates are consistent and asymptotically normal, though in small samples this approximation appears to be unreliable (see later).

A general method for comparing the fit of nested models fitted by REML is the REML likelihood ratio test, or REMLRT. The REMLRT is only valid if the fixed effects are the same for both models. In ASReml-R this requires not only the same fixed effects model, but also the same parameterisation.

If ℓ_{R2} is the REML log-likelihood of the more general model and ℓ_{R1} is the REML log-likelihood of the restricted model (that is, the REML log-likelihood under the null hypothesis), then the REMLRT is given by

$$D = 2\log(\ell_{R2}/\ell_{R1}) = 2\left[\log(\ell_{R2}) - \log(\ell_{R1})\right] \tag{2.20}$$

which is strictly positive. If r_i is the number of parameters estimated in model i, then the asymptotic distribution of the REMLRT, under the restricted model is $\chi^2_{r_2-r_1}$.

The REMLRT is implicitly two-sided, and must be adjusted when the test involves an hypothesis with the parameter on the boundary of the parameter space. It can be shown that for a single variance component, the theoretical asymptotic distribution of the REMLRT is a mixture of χ^2 variates, where the mixing probabilities are 0.5, one with 0 degrees of freedom (spike at 0) and the other with 1 degree of freedom. The approximate P-value for the REMLRT statistic (D), is $0.5(1\text{-Pr}(\chi^2_1 \leq d))$ where d is the observed value of D. This has a 5% critical value of 2.71 in contrast to the 3.84 critical value for a χ^2 variate with 1 degree of freedom. The distribution of the REMLRT for the test that k variance components are zero, or tests involved in random regressions,

which involve both variance and covariance components, involves a mixture of χ^2 variates from 0 to k degrees of freedom. See Self and Liang (1987) for details.

Tests concerning variance components in generally balanced designs, such as the balanced one-way classification, can be derived from the usual analysis of variance. It can be shown that the REMLRT for a variance component being zero is a monotone function of the F statistic for the associated term.

To compare two (or more) non-nested models we can evaluate the *Akaike Information Criteria* (AIC) or the *Bayesian Information Criteria* (BIC) for each model. These are given by

$$\begin{aligned} \mathsf{AIC} &= -2\ell_{Ri} + 2t_i \\ \mathsf{BIC} &= -2\ell_{Ri} + t_i \log \nu \end{aligned} \tag{2.21}$$

where t_i is the number of variance parameters in model i and $\nu = n - p$ is the residual degrees of freedom. AIC and BIC are calculated for each model and the model with the smallest value is chosen as the preferred model.

2.4.2 Diagnostics

In this section we will briefly review some of the diagnostics that have been implemented in ASReml-R for examining the adequacy of the assumed variance matrix for either R or G structures, or for examining the distributional assumptions regarding $\boldsymbol{e}$ or $\boldsymbol{u}$. Firstly we note that the E-BLUP of the residual vector is given by

$$\begin{aligned} \tilde{\boldsymbol{e}} &= \boldsymbol{y} - \boldsymbol{W}\tilde{\boldsymbol{\beta}} \\ &= \boldsymbol{R}_v \boldsymbol{P} \boldsymbol{y} \end{aligned} \tag{2.22}$$

It follows that

$$\begin{aligned} \mathrm{E}\left(\tilde{\boldsymbol{e}}\right) &= \boldsymbol{0} \\ \mathrm{var}\left(\tilde{\boldsymbol{e}}\right) &= \boldsymbol{R}_v - \boldsymbol{W}\boldsymbol{C}^{-1}\boldsymbol{W}^{\mathsf{T}} \end{aligned}$$

The matrix $\boldsymbol{W}\boldsymbol{C}^{-1}\boldsymbol{W}^{\mathsf{T}}$ (under the sigma parameterization) is the so-called 'extended hat' matrix. ASReml-R includes the σ^2 in the hat matrix under the gamma parameterization. It is the linear mixed effects model analogue of $\sigma^2\boldsymbol{X}(\boldsymbol{X}^{\mathsf{T}}\boldsymbol{X})^{-1}\boldsymbol{X}^{\mathsf{T}}$ for ordinary linear models. The diagonal elements are returned in the `hat` component of the asreml object.

If aom = T in the call to asreml() (aom=T can also be set in asreml.options()), ASReml-R returns

- $\boldsymbol{G}^{-1}\tilde{\boldsymbol{u}}$ and $\boldsymbol{G}^{-1}\tilde{\boldsymbol{u}}/\mathrm{diag}\sqrt{\boldsymbol{G}^{-1} - \boldsymbol{G}^{-1}\boldsymbol{C}^{ZZ}\boldsymbol{G}^{-1}}$ as a two column matrix
- $\boldsymbol{R}_v^{-1}\tilde{\boldsymbol{e}}$ and $\boldsymbol{R}_v^{-1}\tilde{\boldsymbol{e}}/\mathrm{diag}\sqrt{\boldsymbol{R}_v^{-1} - \boldsymbol{R}_v^{-1}\boldsymbol{W}\boldsymbol{C}^{-1}\boldsymbol{W}^{\mathsf{T}}\boldsymbol{R}_v^{-1}}$ as a two column matrix.

Note aom = T has not been validated for multivariate models or XFA models with zero specific variances.

The variogram has been suggested as a useful diagnostic for assisting with the identification of appropriate variance models for spatial data (Cressie, 1991). Gilmour *et al.* (1997) demonstrate

its usefulness for the identification of the sources of variation in the analysis of field experiments. If the elements of the data vector (and hence the residual vector) are indexed by a vector of spatial coordinates, $\boldsymbol{s}_i, i = 1, \ldots, n$, then the ordinates of the sample variogram are given by

$$v_{ij} = \frac{1}{2}\left[\tilde{e}_i(\boldsymbol{s}_i) - \tilde{e}_j(\boldsymbol{s}_j)\right]^2, \quad i, j = 1, \ldots, n; \;\; i \neq j$$

The sample variogram is calculated from the triple $(l_{ij1}, l_{ij2}, v_{ij})$ where $l_{ij1} = |s_{i1} - s_{j1}|$ and $l_{ij2} = |s_{i2} - s_{j2}|$ are the absolute displacements. As there will be many v_{ij} with the same displacements, ASReml-R calculates the means for each absolute displacement pair l_{ij1}, l_{ij2} in which case plot.asreml displays the vector $(l_{ij1}, l_{ij2}, \bar{v}_{ij})$ as a perspective plot of the one or two surfaces indexed by the absolute displacement group. In this case, the two directions may be on different scales.

If the coordinates do not form a complete lattice, the `varioGram()` method can be used to form variograms based on polar coordinates. Given a coordinate system (x, y), a response vector z (from the resid() method, say), a vector of directions and a strategy for binning distances, asreml_varioGram() will return a data frame of variogram estimates indexed by direction and distance suitable for a trellis plot.

ASReml-R also computes the variogram from predictors of random effects which appear to have a variance structure defined in terms of distance. The coordinates can be accessed using the varioGram() function, see the ASReml-R Package Reference available at http://asreml.org under Resources > ASReml docs and on the VSN International website https://www.vsni.co.uk, for details.

2.5 Inference: Fixed effects

2.5.1 Introduction

Inference for fixed effects in linear mixed models introduces some difficulties. In general, the methods used to construct F-tests in analysis of variance and regression cannot be used for the diversity of applications of the general linear mixed model available in ASReml-R. One approach would be to use likelihood ratio methods such as Welham and Thompson (1997) although their approach is not easily implemented.

Wald-type test procedures are generally favoured for conducting tests concerning $\boldsymbol{\tau}$. The traditional Wald statistic to test the hypothesis $H_0: \; \boldsymbol{L\tau} = \boldsymbol{l}$ for given $\boldsymbol{L}$, $r \times p$, and $\boldsymbol{l}$, $r \times 1$, is given by

$$\mathcal{W} = (\boldsymbol{L\hat{\tau}} - \boldsymbol{l})^{\mathsf{T}}\{\boldsymbol{L}(\boldsymbol{X}^{\mathsf{T}}\boldsymbol{H}^{-1}\boldsymbol{X})^{-1}\boldsymbol{L}^{\mathsf{T}}\}^{-1}(\boldsymbol{L\hat{\tau}} - \boldsymbol{l}) \tag{2.23}$$

and asymptotically, this statistic has a chi-square distribution on r degrees of freedom. These are marginal tests, so that there is an adjustment for all other terms in the fixed part of the model. It is also anti-conservative if p-values are constructed because it assumes the variance parameters are known.

The small sample behaviour of such statistics has been considered by Kenward and Roger (1997) in some detail. They presented a scaled Wald statistic, together with an F-approximation to its sampling distribution which they showed performed well in a range (though limited in terms of the range of variance models available in ASReml-R) of settings.

2.5 Inference: Fixed effects

In the following we describe the facilities currently available in ASReml-R for conducting inference concerning terms which are in the dense fixed effects model component of the general linear mixed model. These facilities are not available for any terms in the sparse model. These include facilities for computing two types of Wald statistics and partial implementation of the Kenward and Roger adjustments.

2.5.2 Incremental and Conditional Wald Statistics

The basic tool for inference is the Wald statistic defined in equation 14.1. However, there are several ways $\boldsymbol{L}$ can be defined to construct a test for a particular model term, two of which are available in ASReml-R. An F-statistic is obtained by dividing the Wald statistic by r, the numerator degrees of freedom. In this form it is possible to perform an approximate F test if we can deduce the denominator degrees of freedom. For balanced designs, these Wald F statistics are numerically identical to the F-tests obtained from the standard analysis of variance.

The first method for computing Wald statistics (for each term) is the *incremental* form. For this method, Wald statistics are computed from an incremental sum of squares in the spirit of the approach used in classical regression analysis (see Searle; 1971). For example, if we consider a very simple model with terms relating to the main effects of two qualitative factors A and B, given symbolically by

$$\mathsf{y} \sim 1 + \mathsf{A} + \mathsf{B}$$

where 1 represents the constant term (μ), then the incremental sums of squares for this model can be written as the sequence

$$\begin{aligned} R(1) & \\ R(\mathsf{A}|1) &= R(1,\mathsf{A}) - R(1) \\ R(\mathsf{B}|1,\mathsf{A}) &= R(1,\mathsf{A},\mathsf{B}) - R(1,\mathsf{A}) \end{aligned}$$

where the $R(\cdot)$ operator denotes the reduction in the total sums of squares due to a model containing its argument and $R(\cdot|\cdot)$ denotes the difference between the reduction in the sums of squares for any pair of (nested) models. Thus $R(\mathsf{B}|1, A)$ represents the difference between the reduction in sums of squares between the *maximal* model

$$\mathsf{y} \sim 1 + \mathsf{A} + \mathsf{B}$$

and

$$\mathsf{y} \sim 1 + \mathsf{A}$$

Implicit in these calculations is that

- we only compute Wald statistics for *estimable* functions (Searle; 1971, p 408)
- all variance parameters are held fixed at the current REML estimates from the maximal model

In this example, it is clear that the incremental Wald statistics may not produce the *desired* test for the main effect of A, as in many cases we would like to produce a Wald statistic for A based on

$$R(\mathsf{A}|1,\mathsf{B}) = R(1,\mathsf{A},\mathsf{B}) - R(1,\mathsf{B})$$

2.5 Inference: Fixed effects

The issue is further complicated when we invoke *marginality* considerations. The issue of marginality between terms in a linear (mixed) model has been discussed in much detail by Nelder (1977). In this paper Nelder defines marginality for terms in a factorial linear model with qualitative factors, but later (Nelder; 1994) extended this concept to functional marginality for terms involving quantitative covariates and for mixed terms which involve an interaction between quantitative covariates and qualitative factors. Referring to our simple illustrative example above, with a full factorial linear model given symbolically by

$$y \sim 1 + A + B + A.B$$

then A and B are said to be marginal to A.B, and 1 is marginal to A and B. In a three way factorial model given by

$$y \sim 1 + A + B + C + A.B + A.C + B.C + A.B.C$$

the terms A, B, C, A.B, A.C and B.C are marginal to A.B.C. Nelder (1977, 1994) argues that meaningful and interesting tests for terms in such models can only be conducted for those tests which respect marginality relations. This philosophy underpins the following description of the second Wald statistic available in ASReml-R, the so-called *conditional* Wald statistic. This method is invoked by specifying ssType = conditional in wald.asreml(). ASReml-R attempts to construct conditional Wald statistics for each term in the fixed dense linear model so that marginality relations are respected. As a simple example, for the three way factorial model the conditional Wald statistics would be computed as

Term	Sums of Squares		M code
1	R(1)		.
A	R(A | 1,B,C,B.C)	= R(1,A,B,C,B.C) - R(1,B,C,B.C)	A
B	R(B | 1,A,C,A.C)	= R(1,A,B,C,A.C) - R(1,A,C,A.C)	A
C	R(C | 1,A,B,A.B)	= R(1,A,B,C,A.B) - R(1,A,B,A.B)	A
A.B	R(A.B | 1,A,B,C,A.C,B.C)	= R(1,A,B,C,A.B,A.C,B.C) - R(1,A,B,C,A.C,B.C)	B
A.C	R(A.C | 1,A,B,C,A.B,B.C)	= R(1,A,B,C,A.B,A.C,B.C) - R(1,A,B,C,A.B,B.C)	B
B.C	R(B.C | 1,A,B,C,A.B,A.C)	= R(1,A,B,C,A.B,A.C,B.C) - R(1,A,B,C,A.B,A.C)	B
A.B.C	R(A.B.C | 1,A,B,C,A.B,A.C,B.C)	= R(1,A,B,C,A.B,A.C,B.C,A.B.C) - R(1,A,B,C,A.B,A.C,B.C)	C

Of these the conditional Wald statistic for the 1, B.C and A.B.C terms would be the same as the incremental Wald statistics produced using the linear model

$$y \sim 1 + A + B + C + A.B + A.C + B.C + A.B.C$$

The preceding table includes a *marginality* or M code reported when conditional Wald statistics are requested. All terms with the highest M code letter are tested conditionally on all other terms in the model, that is, by dropping the term from the maximal model. All terms with the preceding M code letter, are marginal to at least one term in a higher group, and so forth. For example, in the table, model term A.B has M code B because it is marginal to model term A.B.C and model term A has M code A because it is marginal to A.B, A.C and A.B.C. Model term mu (M code .) is a special case in that it is marginal to factors in the model but not to covariates.

Consider now a nested model which might be represented symbolically by

$$y \sim 1 + REGION + REGION.SITE$$

For this model, the incremental and conditional Wald tests will be the same. However, it is not uncommon for this model to be specified as

$$y \sim 1 + REGION + SITE$$

with SITE identified across REGION rather than within REGION. Then the nested structure is hidden but ASReml-R will still detect the structure and produce a valid conditional Wald F-statistic. This situation will be flagged in the M code field by changing the letter to lower case. Thus, in the nested model, the three M codes would be ., A and B because REGION.SITE is obviously an interaction dependent on REGION. In the second model, REGION and SITE appear to be independent factors so the initial M codes are ., A and A. However they are not independent because REGION removes additional degrees of freedom from SITE, so the M codes are changed from ., A and A to ., a and A.

We advise users that the aim of the conditional Wald statistic is to facilitate inference for fixed effects. It is not meant to be prescriptive nor is it foolproof for every setting.

The Wald statistics are collectively returned by wald.asreml(). The basic table includes the numerator degrees of freedom (denoted ν_{1i}) and the incremental Wald F-statistic for each term. To this is added the conditional Wald F-statistic and the M code if ssType="conditional".

2.5.3 Kenward and Roger Adjustments

In moderately sized analyses, ASReml-R can also calculate the denominator degrees of freedom (DenDF, denoted by ν_{2i}, (Kenward and Roger; 1997)) and a probability value if these can be computed. They will be for the conditional Wald F-statistic if it is reported. The denDF argument of wald.asreml() controls the suppression (denDF = "none") or the use of a particular algorithmic method: denDF = "numeric" for numerical derivatives or denDF = "algebraic" for algebraic derivatives. The value in the probability column is computed from an $F_{\nu_{1i},\nu_{2i}}$ reference distribution. When the DenDF is not available, it is possible, though anti-conservative, to use the residual degrees of freedom for the denominator.

Kenward and Roger (1997) pursued the concept of construction of Wald-type test statistics through an adjusted variance matrix of $\hat{\boldsymbol{\tau}}$. They argued that it is useful to consider an improved estimator of the variance matrix of $\hat{\boldsymbol{\tau}}$ which has less bias and accounts for the variability in estimation of the variance parameters. There are two reasons for this. Firstly, the small sample distribution of Wald tests is simplified when the adjusted variance matrix is used. Secondly, if measures of precision are required for $\hat{\boldsymbol{\tau}}$ or effects therein, those obtained from the adjusted variance matrix will generally be preferred. Unfortunately the Wald statistics are currently computed using an unadjusted variance matrix.

2.5.4 Approximate stratum variances

The svc method returns approximate stratum variances and degrees of freedom for simple variance components models.

For the linear mixed-effects model with variance components (setting $\sigma^2_H = 1$) where $\boldsymbol{G} = \oplus_{j=1}^{q} \gamma_j \boldsymbol{I}_{b_j}$, it is often possible to consider a natural ordering of the variance component parameters including σ^2. Based on an idea due to Thompson (1980) ASReml-R computes approximate stratum degrees of freedom and stratum variances by a modified Cholesky diagonalisation of the expected (or average) information matrix. That is, if $\boldsymbol{F}$ is the average information matrix for $\boldsymbol{\sigma}$, let $\boldsymbol{U}$ be an upper triangular matrix such that $\boldsymbol{F} = \boldsymbol{U}'\boldsymbol{U}$. Further we define

$$\boldsymbol{U}_c = \boldsymbol{D}_c \boldsymbol{U}$$

where $\boldsymbol{D}_c$ is a diagonal matrix whose elements are given by the inverse elements of the last column of $\boldsymbol{U}$ ie $d_{cii} = 1/u_{ir}, i = 1, \ldots, r$. The matrix $\boldsymbol{U}_c$ is therefore upper triangular with the elements in the last column equal to one. If the vector $\boldsymbol{\sigma}$ is ordered in the *natural* way, with σ^2 being the last element, then we can define the vector of so called *pseudo* stratum variance components by

$$\boldsymbol{\xi} = \boldsymbol{U}_c \boldsymbol{\sigma}$$

Thence

$$\text{var}\,(\boldsymbol{\xi}) = \boldsymbol{D}_c^2$$

The diagonal elements can be manipulated to produce effective stratum degrees of freedom (Thompson; 1980) viz

$$\nu_i = 2\xi_i^2 / d_{cii}^2$$

In this way the closeness to an orthogonal block structure can be assessed.

3 Fitting the mixed model

This chapter begins with a brief introduction covering data frame preparation, fitting the linear model and the fitted asreml object followed by a detailed description of the asreml() function call and some technical details of model fitting including the treatment of missing values and setting initial values for variance parameters. The basic concepts are illustrated using a real example and pointers to following chapters are given. For consistency, the same data are also used for illustration in later chapters where possible.

Advanced topics such as models for variance components or genetic models are considered in later chapters. Chapter 7 gives a lengthy set of additional worked examples.

3.1 The data frame

Data for analysis using ASReml-R are generally contained in a text file or a spreadsheet and are read into a *data frame* using the appropriate R functions. Variates and factors in the data frame are then resolved through the data argument of the asreml() function call.

The first 25 lines of the comma separated text file `nin89.csv` containing the NIN field trial data described in Section 1.3.1 are reproduced below. Note that the data are in field order (rows within columns) and a header line (first row) is included. In this case there are 11 comma separated data fields (Variety. . . Column) and the complete file has 224 data rows, one for each variety in each replicate.

```
Variety,ID,pid,raw,Replicate,nloc,yield,lat,long,Row,Column
LANCER,1,1101,585,1,4,29.25,4.3,19.2,16,1
BRULE,2,1102,631,1,4,31.55,4.3,20.4,17,1
REDLAND,3,1103,701,1,4,35.05,4.3,21.6,18,1
CODY,4,1104,602,1,4,30.1,4.3,22.8,19,1
ARAPAHOE,5,1105,661,1,4,33.05,4.3,24,20,1
NE83404,6,1106,605,1,4,30.25,4.3,25.2,21,1
NE83406,7,1107,704,1,4,35.2,4.3,26.4,22,1
NE83407,8,1108,388,1,4,19.4,8.6,1.2,1,2
CENTURA,9,1109,487,1,4,24.35,8.6,2.4,2,2
SCOUT66,10,1110,511,1,4,25.55,8.6,3.6,3,2
COLT,11,1111,502,1,4,25.1,8.6,4.8,4,2
NE83498,12,1112,492,1,4,24.6,8.6,6,5,2
NE84557,13,1113,509,1,4,25.45,8.6,7.2,6,2
```

3.1 The data frame

```
NE83432,14,1114,268,1,4,13.4,8.6,8.4,7,2
NE85556,15,1115,633,1,4,31.65,8.6,9.6,8,2
NE85623,16,1116,513,1,4,25.65,8.6,10.8,9,2
CENTURAK78,17,1117,632,1,4,31.6,8.6,12,10,2
NORKAN,18,1118,446,1,4,22.3,8.6,13.2,11,2
KS831374,19,1119,684,1,4,34.2,8.6,14.4,12,2
⋮
```

This is typical of the required format: a matrix of observations with a row for each sampling unit and columns containing variates, covariates, factors, weights and identities in any convenient order. An optional, though recommended, header line can be used to name the data columns and missing values are denoted by NA.

A data frame is normally created from a text file source using an R function call like:

```
> nin89 <- read.table(file = "nin89.csv", header = T, sep = ",")
```

Consult the R documentation for a detailed description of importing data but some general points to note are:

- blank lines are ignored.
- it is sensible to include a header line in the data file; if no header line is included, the columns are labelled V1...Vn where n is the number of columns.
- the same column label should not be repeated. The numerals 1, 2, etc are appended to subsequent repeated column labels.
- NA is the only acceptable code for missing values.
- in comma separated text (.csv) files
  - consecutive commas imply a missing value.
  - provided the number of fields is consistent, a line beginning (ending) with a comma will generate NA for that observation in the first (last) variate or a zero length string if a text field.
- blanks may be embedded in text fields provided the field delimiter is not also the space character, otherwise the string must be enclosed in quotes.
- too many or too few data fields on a line cause an error.

Character fields such as Variety above are automatically converted to factors with read.table(). However, numeric fields such as Replicate remain as variates so that the user must manually convert numeric fields into factors as required. The utility function asreml.read.table() offers a convenient alternative; asreml.read.table() reads data from a text file and automatically converts variates whose

names begin with a capital letter in the header line into factors. So, for the NIN data

```
> nin89 <- asreml.read.table(file = "nin89.csv", header = T, sep = ",")
```

creates a data frame in which pid, raw, nloc, yield, lat and long are variates, but Variety, ID, Replicate, Row and Column are factors. This is equivalent to the sequence

```
> nin89 <- read.table(file = "nin89.csv", header = T, sep = ",")
> nin89$ID <- factor(nin89$ID)
> nin89$Replicate <- factor(nin89$Replicate)
> nin89$Row <- factor(nin89$Row)
> nin89$Column <- factor(nin89$Column)
```

3.2 Introducing the asreml() function call

A complete asreml() function call for a simple randomised complete block (RCB) analysis of the NIN yield data is

```
> nin89.asr <- asreml(fixed = yield ~ Variety, random = ~idv(Replicate), residual =
    ~idv(units), na.action = na.method(x = "include"), data = nin89)
```

where nin89.asr is the name we have chosen for the returned object. The key elements of this call are outlined below while the components of the returned object are described in Section 3.3.

3.2.1 Model formulae: specifying the linear mixed model

The linear model is specified in the fixed (required), random (optional) and residual (error component) arguments as formula objects. A third optional model argument sparse is also available but is not used explicitly (see also Section 3.10) in this example.

The fixed terms in the model are specified as a formula with the response on the left of a $\sim$ operator and the terms separated by + operators on the right. In this case Variety is a fixed factor in a model for the response variate yield so that the fixed argument is given as

```
> nin89.asr <- asreml(fixed = yield ~ Variety, ...)
```

There must be at least one fixed effect in the model and the response may only be specified in the fixed argument. So, if the intercept was the only fixed term in the model then the fixed argument would be

```
> nin89.asr <- asreml(fixed = yield ~ 1, ...)
```

The random terms in the model are specified as a formula. However, unlike the fixed formula there is no response on the left of the $\sim$ operator. In this example Replicate is a random term so the random argument is

```
> nin89.asr <- asreml(..., random = ~idv(Replicate), ...)
```

The residual or error component of the model is specified in a formula object through the residual

argument. The default is a simple error term representing independent and identically distributed (IID) effects and does not need to be formally specified. However, a special factor units defined as factor(seq(1,n)) where n is the number of observations, is always automatically generated by asreml(), so that the default error model in this case could be specified explicitly in the call

```
> nin89.asr <- asreml(..., residual = ~idv(units), ...)
```

3.2.2 Finding the data

The data argument to asreml() is an optional, though strongly recommended, argument that identifies a data frame containing the variables named in the model specification. The data frame is nin89 in this case. If the data argument is missing then ASReml-R attempts to obey the usual rules for resolving variate names, however, this is not always possible in complex situations with certain special model functions.

3.3 Components of the fitted model: the asreml object

A call to asreml() produces an object of class asreml which contains numerous components of the fit including

- the REML log-likelihood
- best linear unbiased predictors (BLUPs) of the random effects
- generalised least squares estimates of the fixed effects
- REML estimates of variance components
- (optionally) part of the inverse coefficient matrix
- the inverse of the average information matrix
- residuals and fitted values from the linear model.

A complete description of the components of an asreml object is given in the ASReml-R Package Reference which is a pdf version of the ASReml-R help pages obtained in R by typing `help(asreml)`. This document is available at http://asreml.org under `Resources` > `ASReml docs` and on the VSN International website https://www.vsni.co.uk.

3.3.1 Methods and related functions

Specific instances of the standard extractor functions coef(), resid() and fitted() exist, as do summary(), plot() and predict() (see Chapter 6) methods. An anova type method is implemented by wald() (see Section 3.14).

The summary.asreml() function returns a list with a range of components. The variance components are returned in

```
> summary(nin89.asr)$varcomp
```

and the coefficients from the fixed, random and sparse parts of the model are summarised in the coef.fixed, coef.random and coef.sparse components. For example, the fixed effects for Variety are given by

```
> summary(nin89.asr, coef = TRUE)$coef.fixed
```

3.4 A note on data order

The observations must be presented in the order specified by the error model, that is, the value of the residual argument. The assumption of separability is implicit in the use of the colon operator (:). Furthermore, the sort order outer:inner of the observations is implied by the order of appearance of the factors in the residual formula. In the case, for example, where

```
> residual = ~ar1v(Column):ar1(Row)
```

the data is assumed to be sorted as rows within columns. Note that if the sort order of observations is incorrect an error is generated.

3.5 Getting help

A complete description of the asreml object is given in the ASReml-R Package Reference, see above for details, and can be obtained from the help system within R:

```
> `?`(asreml)
```

or

```
> help(asreml)
```

generates text based help or html help depending on the platform and help system state.

3.6 Fixed terms

3.6.1 Dense fixed terms

The fixed model formula specifies the response, fixed factors, interactions and covariates for which standard errors and tests of significance are required. These terms may also include those specified by the relevant model functions from Table 3.1. The fixed formula must contain at least one term which may simply be the intercept. By default the intercept is included in the fixed model; for example,

```
> asreml(fixed = y ~ Variety, ...)
```

includes an intercept plus the main effects for Variety. To specify a model with no overall mean, include a -1 after $\sim$ in the list of primary fixed terms, for example, use

3.6 Fixed terms

```
> asreml(fixed = y ~ -1 + Variety, ...)
```

An intercept-only fixed model is specified by including a 1 only after ∼ , for example,

```
> asreml(fixed = y ~ 1, random = ...)
```

Terms can be modified or generated by special model functions such as lin(). For example, to include a linear (single degree of freedom) effect of Row (a factor with 22 levels) use

```
> asreml(fixed = y ~ lin(Row) + ...)
```

Model functions also exist to generate orthogonal polynomials (pol() and leg()) and to fit terms conditionally (at(); Table 3.1 and Section 3.8). Note that fixed is the only model formula where the response may be specified.

Table 3.1: Summary of reserved names and special functions with their typical usage; fixed (f) or random (r)

term	purpose	usage
reserved names		
mv	fits missing values as covariates. An example of its use is in spatial analyses, for example, where computing advantages arising from a balanced spatial layout can be exploited. Missing values in the response are handled in two ways using the `na.method()` function. If na.action = na.method(y="omit"), records containing missing values in the response are deleted. If na.action = na.method(y="include"), missing values are estimated and a factor labelled `mv` is included in the model frame. If a variate labelled `mv` already exists in the data frame it will be overwritten. For a multivariate analysis, missing values must currently be included	f
trait	used with multivariate data to fit the individual trait means. It is interacted with other factors to estimate their effects for all traits. It is formally equivalent to the intercept (1) but is a more natural label for use with multivariate data. If a variate labelled trait already exists in the data frame it will be overwritten.	f, r
units	a factor with a level for each experimental unit; allows a second error term to be explicitly fitted.	r
model functions		
at(f,l)	condition on level $l = 1, \ldots, k$ of factor f. That is, defines a binary variable which is 1 if the factor f has level l for the observation. For example, to fit a row factor only for site 3, use the expression at(site,3):row. Note that if l is numeric, then the level of f is chosen as the lth in factor (sorted) order. Note also that when used with spline terms, such as at(f,2):spl(x) then the knot points are derived from **all** of factor f, **not** just level 2.	f, r
dev(x)	forms a factor with a level for each unique value of x.	r

3.6 Fixed terms

Summary of reserved names special functions

term	purpose	usage
gpf(obj)	forms a new factor (obj) from an existing factor by merging a subset of its levels. The name obj must also appear as a component of the combine argument to asreml() where the existing factor and the levels to merge are defined with a call to 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. 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 gpf() can be duplicated outside asreml(), gpf() is necessary if the asreml method predict() is to be used.	f, r
grp(obj)	Groups contiguous columns of data to be treated as a single factor named "obj". The columns of data are identified by a character or numeric vector component obj of the group argument to asreml().	r
leg(x,t)	forms t legendre polynomials from the values in x; the mean is excluded if t is negative. For example, leg(time,2) is a factor with three columns: a constant in the first, centred and scaled linear covariate in the second and centred and scaled quadratic covariate in the third. leg() could be interacted with a design factor to fit random regression models.	f, r
lin(f)	treats the named factor as a variate. The function is defined for f being a simple factor, trait and units. The lin(f) function does not center or scale the variable.	f, r
link(a,b)	ensures that the structures for terms a and b are contiguous. The function would typically be used in random coefficient regression, where a covariance between intercept and slope might be required.	r
mbf(obj)	Includes obj as a set of covariates to be fitted as a single term in a similar way to grp. The name obj must also appear as a component of the mbf argument to asreml() where the data frame holding the covariates is identified along with a key field for merging records with those in data.	r
pol(x,t)	forms t orthogonal polynomials from the values in x; the mean is excluded if t is negative. For example, pol(time,2) is a factor with three columns: a constant in the first, centred and scaled linear covariate in the second and centred and scaled quadratic covariate in the third. pol() could be interacted with a design factor to fit random regression models.	f, r
sbs(obj)	forms a new factor (obj) from an existing factor by selecting a subset of its levels. The name obj must also appear as a component of the prune argument to asreml() where the existing factor and the subset of levels to select are defined with a call to Subset(). 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. 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 sbs() can be duplicated outside asreml(), sbs() is necessary if the asreml method predict() is to be used.	f, r
spl(x, k)	Random component of a cubic spline for covariate x. spl(x), dev(x) and possibly lin(x) are used when fitting cubic splines. The cubic spline is composed of a random nonlinear component imposed on a linear trend. It is fitted by including a special random factor, spl(x), and the fixed covariate (x) in the linear model. Knot points are placed at the design points if length(unique(x)) $< k$ otherwise there are k equally spaced knot points over the range of x. The default for k is 50. If k is omitted then knots can be set in asreml.options(). Also, explicit knot points are set in the knot.points argument to asreml().	r

Summary of reserved names special functions

term	purpose	usage

3.6.2 Sparse fixed terms

The sparse argument specifies those covariates, factors and interactions for which standard errors and tests of significance are not required. These effects are estimated using sparse matrix methods that typically require less memory and less execution time. ASReml-R automatically includes missing values in the sparse component with a factor named mv. This is a reserved word and should not be used to label variates or factors.

3.6.3 Covariates

For analysis purposes it is recommended that covariates be centred or rescaled to have a variance of 1 to avoid failure to detect singularities. In addition, missing values in covariates are replaced with zeros so it is important in these circumstances to centre the covariate in question. For example, the command

```
> nin89$linrow <- as.numeric(nin89$Row) - mean(as.numeric(nin89$Row), na.rm = T)
```

could be used to create a mean centred row covariate. Care should also be exercised when scaling variates for use in random coefficient or spline models.

3.7 Random terms

The random model formula specifies the factors, interactions, covariates and special terms that comprise the random component of the model. These effects are estimated using sparse matrix methods. Each random term will have a variance model associated with it which, when no variance model function is specified, defaults to a scaled identity $\gamma \boldsymbol{I}_n$ or $\sigma^2 \boldsymbol{I}_n$ where γ is a variance ratio, depending on whether a sigma or gamma parameterization is used for fitting. See page 63 under Rules for combining variance models.

3.7.1 Initial values and constraints for variance parameters

Initial values and constraints for variance parameters are held in list objects that represent the structure of the error variance matrix (referred to as R structures in this manual and denoted $\boldsymbol{R}$ algebraically, see Chapter 4) and the variance matrix for the other random terms in the model (referred to as G structures and denoted $\boldsymbol{G}$ algebraically). Initial values are for the parameters being fitted so depend on the parameterization used. The default initial values are 0.1 for variance ratios (when parameters are estimated using the gamma parameterization) and 0.1*v for variance components (when parameters are estimated using the sigma parameterization), where v is half the simple variance of the response. Using both parameterizations correlations are assumed to have a starting value of 0.1. The corresponding default parameter constraints are P (positive) for variance

component ratios, U (unconstrained) for correlations and P for variance components.

For example, in the simple RCB field trial analysis

```
> asreml(fixed = yield ~ Variety, random = ~idv(Replicate), residual = ~id(units), data =
    nin89)
```

the gamma parameterization is used and a single variance component ratio is estimated for the random Replicate term using an initial starting value of 0.1 and default constraint of P (that is, the parameter is constrained to be positive).

The default starting values and boundary constraints may not be either adequate or appropriate in all circumstances. There are two ways to alter the starting values and constraints from their default state, both of which rely on exporting the internally generated names of the variance components along with their values and constraints to an R object or external text file. The G.param and R.param arguments are used to subsequently overwrite the default initial values and constraints in an analysis. An initial value object is created by setting the start.values argument to asreml().

3.7.1.1 Replacing elements in an internal object

As an example, to set a different initial value for the Replicate component, the call

```
> nin89.sv <- asreml(fixed = yield ~ Variety, random = ~idv(Replicate), residual =
    ~id(units), na.action = na.method(x = "include"), data = nin89, start.values = TRUE)
```

returns a list object nin89.sv with components G.param, R.param and vparameters.table. The first two components are list objects while vparameters.table is a data frame containing the parameter names, their initial values and boundary constraints.

```
> iv <- nin89.sv$vparameters.table
> iv
```

Elements of this table can be set by the usual R replacement methods. The new initial value for Replicate can be used in asreml() with the G.param argument. That is,

```
> nin89.asr <- asreml(fixed = yield ~ Variety, random = ~idv(Replicate), residual =
    ~id(units), na.action = na.method(x = "include"), data = nin89, G.param = iv)
```

3.7.1.2 Editing an external text file

An alternative is to specify a filename as the value of the start.values argument. This creates a comma separated text file version of vparameters.table, with a header line and columns containing the component name and its initial state. After editing this file, the revised initial values or constraints can be used similarly to the above by specifying the text file name as the value of the G.param argument. For example, the following call creates a comma separated textfile (`filename`) for editing

```
> nin89.sv <- asreml(fixed = yield ~ Variety, random = ~idv(Replicate), residual =
    ~idv(units), na.action = na.method(x = "include"), data = nin89, start.values =
```

```
    "filename")
```

with the revised values included in the analysis by:

```
> nin89.sv <- asreml(fixed = yield ~ Variety, random = ~idv(Replicate), residual =
    ~idv(units), na.action = na.method(x = "include"), data = nin89, G.param = "filename")
```

Note that in the above sequence, a list with components G.param, R.param and vparameters.table is still returned in nin89.sv.

3.7.2 Specifying variance structures

As stated above, the default variance model for a term in the random model is a scaled identity ($\gamma \boldsymbol{I}_n$ or $\sigma^2 \boldsymbol{I}_n$), that is, independent and identically distributed (IID). This is a special case of a more general scaled parameterised matrix. An extensive range of variance models can be fitted to terms in the random formula and error (residual) component of the model. These are specified using special functions in the model formulae and are described in Chapter 4. For example, the experimental units of nin89 are indexed by Column and Row, respectively. If we first augment the data frame to complete the 22 row by 11 column array of plots, we could then specify a separable first order autoregressive process (Gilmour et al.; 1997) in two dimensions by including

```
> residual = ~ar1v(Column):ar1(Row)
```

(assuming the data is correctly ordered as *Row* within *Column*) in the call, where ar1() is a special function specifying a first order autoregressive variance model for both Column and Row, see Section 4.1. The complete range of possible variance models is presented in Table B.1.

The behaviour of these special functions can be different from the expected behaviour of standard R functions; they generally return existing or altered attributes of objects and/or set up internal structures for the model fitting algorithm. There are some restrictions on usage, notably nesting. However, there are few instances where it is sensible to nest these functions, one exception being models with random coefficients.

3.8 Conditional factors: the at() and dsum() functions

A conditional factor is a factor that is present only when another factor has a particular level. For example, in a multi-environment trial analysis over 2 sites where each site is a randomised complete block design, we could estimate separate Block variance components for each Site by including the random term at(Site):Block. If no levels of the conditioning factor (Site in this case) are specified in the at() function, a complete set of conditioning terms is generated. In this example at(Site):Block expands to at(Site,1):Block + at(Site,2):Block. Note that this is also equivalent to fitting a diagonal variance model using diag(Site):Block.

If the levels vector (l) of the conditioning factor (f) is specified as a numeric vector then it refers to the levels of f in the order returned by levels(f). The at() function is only associated with random terms and cannot be wrapped with a variance function as the results can be ambiguous. A similar function dsum() is available for the residual formula and specifies a variance model for $\boldsymbol{e}$ as a direct sum of l variance matrices, one for each level of the conditioning factor.

3.8 Conditional factors: the `at()` and `dsum()` functions

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 data set comprising data for three sites, each laid out in a row by column array coded by factors `Row` and `Column` in the data set. 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

3.8 Conditional factors: the `at()` and `dsum()` functions

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.

3.9 Weights

Weighted analyses are achieved by using the weights = *wt* argument to asreml(), where *wt* is a variate in the data frame. If these are relative weights (to be scaled by the units variance) then this is all that is required; for example, the number of sampling units (wt=c(3, 1, 3, ...)). If they are absolute weights, that is, the reciprocal of known variances, the units variance should be constrained to 1. This can be done by one of two ways:

1. one of the methods described in Section 3.7.1, that is, editing a default R parameter list object with asreml.gammas.ed() (start.values=T) or create and edit an external text file with start.values="filename", changing the constraint of the units variance to F.

2. Set the units variance with the family argument

```
> fm <- asreml(..., family = asr_gaussian(dispersion = 1), ...)
```

3.10 Missing values

Missing values have been included in `nin89.csv` for the convenience of fitting spatial models in subsequent examples. By default, missing values in covariates or factors cause an error: na.action = na.method(x = "fail"). Missing values are treated as follows:

3.10.1 Missing values in the response

Records with missing values in the response are included by default, na.action = na.method(y = "include"), and are estimated as a consequence of fitting the model. A factor labelled mv is created and included in the sparse equations, and the solutions are returned in coef(object)$sparse. An alternative action is "omit" which excludes units with missing values in the response. Missing values must be estimated in a multivariate analysis.

3.10.2 Missing values in the explanatory variables

Covariates Records with missing values in covariates are only discarded if na.action = na.method(x ="omit"). If included, they are treated as zeros which may only be reasonable if the covariate values are centred.

Design factors Missing values are allowed in design factors and handled as for covariates. Where this occurs, no formal level is assigned to the factor for that record, however, the missing value is replaced by a zero in the fitting process.

3.11 Generalized linear models

asreml() includes family functions for fitting Generalized Linear Models (McCullagh and Nelder; 1994). These differ from the standard family functions through the addition of a dispersion argument

which determines whether the dispersion parameter is fixed or estimated (dispersion=NA). Table 3.2 lists the link functions that can be used to connect the linear predictor $\boldsymbol{\eta}$ to the mean ($\boldsymbol{\mu}$) on the data scale.

Table 3.2: Families and link functions

Link	Function	gaussian	binomial	poisson	Gamma
`identity`	$\eta = \mu$	D		*	*
`sqrt`	$\eta = \sqrt{\mu}$			*	
`log`	$\eta = \log(\mu)$	*		D	*
`inverse`	$\eta = 1/\mu$	*			D
`logit`	$\eta = \ln(\mu/(1-\mu))$		D		
`probit`	$\eta = \Phi^{-1}(\mu)$		*		
`cloglog`	$\eta = \log(-\log(1-\mu))$		*		

where μ is the mean on the data scale, $\boldsymbol{\eta} = \boldsymbol{X\tau}$ is the linear predictor on the underlying scale and D is the default.

3.12 Generalized Linear Mixed Models

There is the capacity to fit a wider class of models which include additional random effects for non-normal error distributions. The inclusion of random terms in a GLM is usually referred to as a Generalized Linear Mixed Model (GLMM). For GLMMs, asreml() uses what is commonly referred to as penalized quasi-likelihood or PQL (Breslow and Clayton; 1993). The technique is also known by other names, including Schall's technique (Schall; 1991), pseudo-likelihood (Wolfinger and O'Connell; 1993) and joint maximisation (Harville and Mee; 1984; Gilmour et al.; 1985). It is implemented in many statistical packages, for instance, in the GLMM procedure (Welham; 2005) and the IRREML procedure of Genstat (Keen; 1994), in MLwiN (Goldstein et al.; 1998), in the GLMMIXED macro in SAS and in the GLMMPQL function in R, to name a few.

The PQL technique is based on a first order Taylor series approximation to the likelihood. It has been shown to perform poorly for certain types of GLMMs. In particular, for binary GLMMs where the number of random effects is large compared to the number of observations, it can underestimate the variance components severely (up to 50%) (for example, Breslow and Lin (1995); Goldstein and Rasbash (1996); Rodriguez and Goldman (2001); Waddington et al. (1994)). For other types of GLMMs, such as Poisson data with many observations per random effect, it has been reported to perform quite well (Breslow; 2003, for example). As well as the above references, users can consult McCulloch and Searle (2001) for more information about GLMMs.

Most studies investigating PQL have focussed on estimation bias. Much less attention has been given to the wider inferential issues such as hypothesis testing. In addition, the performance of this technique has only been assessed on a small set of relatively simple GLMMs. Anecdotal evidence from users suggests that this technique can give very misleading results in certain situations.

Therefore, we cannot recommend the use of this technique for general use. It is included in the

current version of asreml() for advanced users. It is highly recommended that its use be accompanied by some concerned. For instance, one way of doing this would be by simulating data using the same design and using parameter values similar to the parameter estimates achieved, such as used in Millar and Willis (1999).

3.13 Multivariate analysis

Multivariate analysis is used when we are interested in estimating the correlations between distinct traits (for example, fleece weight and fibre diameter in sheep) and for repeated measures of a single trait. The term *multivariate* analysis is used here in the narrow sense of a multivariate mixed model. There are many other multivariate analysis techniques which are not covered by asreml().

3.13.1 Model specification

If the response term specified in the fixed formula of a asreml() call is a matrix then a multivariate analysis is automatically performed. That is, for response variates $y_1, \ldots, y_k$ in the data frame, a multivariate analysis would be specified with the call

```
> asreml(fixed = cbind(y_1, ..., y_k) ~ trait, ...)
```

In this case, asreml() creates a factor trait (the multivariate equivalent to the univariate general mean) with the names of the response variates as levels.

A multivariate analysis in asreml() can be specified in one of two ways:

- specifying a matrix as the response in the fixed formula, as noted above. For the wether trial data, the term trait is a factor generated by asreml() with $ntr = 2$ levels gfw and fdiam. Internally, asreml() expands the data frame by repeating each row ntr times such that traits are nested within experimental units,

- specifying the asmv=*trait* argument; this assumes that the data frame has been expanded into a univariate form outside asreml(). In this case the order need not necessarily be *traits within units* but the order of terms in the residual formula must reflect the data order. Note that in this case *trait* refers to the factor in the data frame that defines the traits but is not necessarily named trait.

The following examples illustrate the specification of multivariate models in asreml(), some components of the returned object and the wald() method.

3.13.1.1 A repeated measures example

Wolfinger (1996) summarises a range of variance structures that can be fitted to repeated measures data, demonstrating the models using the rat data set described in Section 1.3.2. The asreml() function call for an analysis of the five repeated measures is:

```
> wolfinger.asr <- asreml(fixed = cbind(wt0, wt1, wt2, wt3, wt4) ~ trait * Treatment,
    residual = ~id(units):us(trait, init = rep(0, 15)), maxit = 20, data = wolfinger)
```

The use of rep(0,15) as initial values in the above call signals that in a multivariate analysis reasonable starting values are to be calculated from the phenotypic variance-covariance matrix. The fitted variance components are given by:

```
> summary(wolfinger.asr)$varcomp
```

3.13.1.2 A bivariate example

The asreml() function call for a basic bivariate analysis of the wether trial data described in Section 1.3.3 is:

```
> wether.asr <- asreml(cbind(gfw, fdiam) ~ trait + trait:Year, random = ~us(trait, init =
    c(0.4, 0.3, 1.3)):Team + us(trait, init = c(0.2, 0.2, 2)):Tag, residual =
    ~id(units):us(trait, init = c(0.2, 0.2, 0.4)), data = orange)
```

A trace of the model's convergence is held in the trace component:

```
> wether.asr$trace[, c(1, "final", "constraint")]
```

Final estimates of the variance components are given by summary() (illustrated above) and an analysis of variance calculating the approximate denominator degrees of freedom and conditional F-tests can be obtained by:

```
> wald(wth0.asr, denDF = "default", ssType = "conditional")
```

3.13.2 Specifying multivariate variance structures

A more sophisticated default error structure is required for multivariate analysis in ASReml-R. Using the notation of Chapter 4, consider a multivariate analysis with n_t traits and n units in which the data are ordered traits within units. An algebraic expression for the variance matrix in this case is

$$\boldsymbol{I}_n \otimes \boldsymbol{\Sigma}$$

where $\boldsymbol{\Sigma}^{(n_t \times n_t)}$ is an unstructured variance matrix.

For a standard multivariate analysis

- the error structure must be specified as two-dimensional, with independent units and often an unstructured variance matrix across traits.
  - the residual for this model is therefore residual ~ id(units):us(trait)
  - missing values are allowed and **must** be fitted. asreml() automatically includes the special factor mv in the sparse formula in such cases.
- for the default analysis, that is the response is specified as a matrix, the R structure *must* reflect the data order of *traits within units* which means that the term units must appear before trait in the residual formula.

- variance parameters are variances, not variance ratios.

- the error structure is often specified as an unstructured variance matrix but correlation models may also be used. asreml() attempts to detect such cases and fix or estimate the residual scale parameter accordingly.

  For example, with the Wolfinger data the times are equally spaced so we could fit a first order autoregressive model using:

```
> asreml.options(gammaPar = TRUE)
> wolfinger.asr <- asreml(fixed = cbind(wt0, wt1, wt2, wt3, wt4) ~ trait * Treatment,
    residual = ~id(units):ar1(trait), data = wolfinger)
```

- as noted previously, initial values for the variance matrices are given as the lower triangle of the (symmetric) matrix specified row-wise,

- nominating reasonable initial values can be a problem. By default, asreml() uses half the phenotypic variance in forming initial values.

3.14 Testing of terms: the wald() method

The type of object returned by the wald() method depends on the value of the denDF and ssType arguments.

Incremental F-statistics

If denDF = "none" and ssType = "incremental" (the defaults), an object of class anova containing a table of Wald statistics for fixed effects is returned. Terms in the table are tested sequentially, which means that factors are adjusted for terms higher in the table (or not in the table), but ignoring terms that occur below.

No denominator degrees of freedom is supplied as the reference distribution for each Wald statistic is a χ^2_k where k is the number of nonsingular effects in the term.

Conditional F-statistics

If at least one of denDF or ssType is set to anything other than the default, a data frame object is returned that includes columns for the approximate denominator degrees of freedom or conditional F-statistics depending on the combination of options chosen.

The data frame has 3 styles:

```
Source   df             F_inc  F_con M
Source   df  ddf_inc    F_inc           P_inc
Source   df  ddf_con    F_inc  F_con M  P_con
```

depending on whether conditional F-statistics are reported or whether the denominator degrees of freedom are calculated. See Section 2.5 for more background on the contents of this table.

3.14 Testing of terms: the wald() method

The numerator degrees of freedom for each term is easily determined as the number of non-singular effects involved in the term. However, in general calculation of the denominator degrees of freedom is not trivial. ASReml-R will only attempt the calculation if specifically requested as it requires further iterations of the model (using update.asreml()).

4 Specifying variance structures

This chapter introduces variance model specification in ASReml-R, a complex aspect of the modelling process. To summarise the key concepts:

- the mixed linear model
$$\boldsymbol{y} = \boldsymbol{X}\boldsymbol{\tau} + \boldsymbol{Z}\boldsymbol{u} + \boldsymbol{e}$$
has a residual term
$$\boldsymbol{e} \sim N(\boldsymbol{0},\ \boldsymbol{R}_v(\boldsymbol{\sigma}_r))$$
and random effects
$$\boldsymbol{u} \sim N(\boldsymbol{0},\ \boldsymbol{G}(\boldsymbol{\sigma}_g))$$
where, in the most complex forms
$$\boldsymbol{R}_v = \oplus_i \boldsymbol{R}_{v_i}$$
$$\boldsymbol{G} = \oplus_j \boldsymbol{G}_j$$
and each
$$\boldsymbol{R}_{v_i} = \boldsymbol{R}_{v_i}(\boldsymbol{\sigma}_{r_i})$$
$$\boldsymbol{G}_j = \boldsymbol{G}_j(\boldsymbol{\sigma}_{g_j})$$
where $\boldsymbol{\sigma}_{r_i}$ and $\boldsymbol{\sigma}_{g_j}$ parameterise the respective variance models
- we use the terms *R structure* and *G structure* to refer to the matrices $\boldsymbol{R}_{v_i}$ and $\boldsymbol{G}_j$ above in a syntactic manner, respectively
- R and G structures are typically formed as a direct product of particular variance models
- the order of terms in a direct product must agree with the order of effects in the corresponding model term
- variance models may be correlation matrices or variance matrices with equal or unequal variances on the diagonal. A model for a correlation matrix (eg. ar1()) can be converted to an equal variance form (eg. ar1v()) and to a heterogeneous variance form (eg. ar1h())
- variance components are estimated as gammas (relative to the overall scale parameter, σ^2_e) when the gamma parameterization is used.

Chapter 2 gives theoretical details. We begin this chapter by considering an ordered sequence of variance structures for the NIN variety trial (Section 4.1) as an introduction to variance modelling in practice. We then consider the topics in detail.

Variance models are specified with special model functions in the random and residual formulae. Scaled identity defaults are used if no variance model is explicitly specified. Table B.1 presents the

complete range of variance models available in ASReml-R and details of individual (variance model) function calls are given in Section 4.3. Most of the models listed in Table B.1 are correlation models (id() to agau()) but these are easily generalised to:

- homogeneous variance models by appending a v to the function name, for example, converting id() to idv() to specify IID errors,
- heterogeneous variance models by appending h to the function name, for example, converting id() to idh() to specify independent but heterogeneous errors.

Rules for combining variance models and methods for setting initial values are given in Section 4.5.

4.1 A sequence of structures for the NIN field trial data

By way of introduction, six variance structures of increasing complexity are considered for the NIN field trial data (Section 1.3.1). This is to give a general feel for variance modelling in ASReml-R from a practical perspective and some idea of the types of models that are possible (Table B.1).

This section illustrates:

- changes to $\boldsymbol{u}$ and $\boldsymbol{e}$ and the assumptions regarding the variance of these terms
- the impact this has on the random formula for specifying the G structures for $\boldsymbol{u}$ and the residual formulae for specifying the R structure(s) for the residuals in $\boldsymbol{e}$
- for ease of exposition we first describe the models fitted and reported when the variance models are explicitly specified. For the models in this section an alternative fitting algorithm based on a gamma parameterization has advantages. In this section we indicate how ASReml-R can be easily actioned to use this algorithm. In Section 4.8 we give one of the RCB data model examples in this gamma parameterization, provide some alternative specifications and comment on the summaries. In Section 4.1.2 we also show how some specifications can be reduced.

Model 1: randomised complete block (RCB) analysis - blocks fixed

```
> rcb.asr <- asreml(yield ~ Variety + Block, residual = ~idv(units), data = nin89)
```

uses an IDV variance structure for the residual error term assuming that $e \sim N(\mathbf{0}, \sigma_e^2 \boldsymbol{I}_{224})$. The model is therefore a fixed effect model and involves just one R structure and no G terms.

Model 1a: RCB analysis with G and R structures

```
> rcb.asr <- asreml(yield ~ Variety, random = ~idv(Replicate), residual = ~idv(units), data
    = nin89)
```

The residual is specified as a variance matrix with $\text{var}(e) = \sigma_e^2 \boldsymbol{I}_{224}$ and $\boldsymbol{u}$ is a vector of replicate effects where $\text{var}(u) = \sigma_r^2 \boldsymbol{I}_4$. This model introduces the use of variance model functions in both random and residual formulae to explicitly specify the G and R structures.

Note that when specifying G structures ASReml-R automatically adds a scale parameter if a correlation model is specified (see Section 4.1.2 for more detail). At most one of the models specified in a G structure can be a variance model. If the variance matrix of a term contains several component

matrices then the problem of *identifiability* arises. For example, a random term

```
idv(A):idv(B)
```

with residual `idv(units)` produces a variance matrix of the form $\sigma_a^2\sigma_b^2\boldsymbol{I}_{a.b}$ for A:B where the σ_a^2, σ_b^2 parameters are not identifiable.

Model 2: two-dimensional spatial model with correlation in one direction

```
> sp.asr <- asreml(yield ~ Variety, residual = ~idv(Column):ar1(Row), data = nin89)
```

This call specifies a two-dimensional spatial structure for error but with spatial correlation in the row direction only. In this case $\text{var}(\boldsymbol{e}) = (\sigma_c^2\boldsymbol{I}_{11}) \otimes \boldsymbol{\Sigma}_r$. The R structure is the direct product of two matrices; a scaled identity matrix of order 11 and a autoregressive correlation matrix of order 22 with elements $\{\sigma_{ij}\} = \rho^{|i-j|}$ for plots (in the same column) in rows i and j. Note that:

- the direct product structure is implied by the ":" operator. The order in which factors appear in the residual formula also specifies the order in which the data must be sorted. Because Column is specified before Row, the implication is that the data are in the order *rows within columns*. ASReml-R does not reorder the observations; if the data frame is not in the order specified by residual then an error is generated and it must be reordered outside asreml().
- using a separable model for the R structure implies that the data can be regarded as a matrix or array whose data is indexed by the levels of the factors that represent the rows and columns of this array. In this field trial example these factors are Row and Column, respectively. For this structure to be applicable, the data in this case must be augmented with 18 additional missing values. Variety is arbitrarily coded as LANCER for all of the extra missing plots.
- ASReml-R automatically includes missing values in the sparse component with a factor named mv (see Section 3.10).

Model 2a: two-dimensional spatial model

```
> sp.asr <- asreml(yield ~ Variety, residual = ~ar1v(Column):ar1(Row), data = nin89)
```

This extends model **2** by specifying a first order autoregressive variance model of order 11 for columns (`ar1v()`). The R structure in this case is therefore the direct product of two autoregressive matrices, one a variance matrix and one a correlation matrix, that is, $\text{var}(\boldsymbol{e}) = (\sigma_c^2\boldsymbol{\Sigma}_c) \otimes \boldsymbol{\Sigma}_r$.

Model 2b: two-dimensional spatial model with measurement error

```
> sp.asr <- asreml(yield ~ Variety, random = ~idv(units), residual =
    ~ar1v(Column):ar1(Row), data = nin89)
```

This model includes a factor with $n = 224$ levels in $\boldsymbol{u}$. Since $\boldsymbol{Z} = \boldsymbol{I}$, $\text{var}(\boldsymbol{y}) = \sigma_{un}^2\boldsymbol{I}_{224} + (\sigma_c^2\boldsymbol{\Sigma}_c) \otimes \boldsymbol{\Sigma}_r$. The quantity σ_{un}^2 is the so-called measurement error variance or nugget variance in geostatistics. units is a reserved name that ASReml-R constructs internally as seq(1,nrow(data)).

Model 3: two-dimensional spatial model defined as a G structure

```
> sp.asr <- asreml(yield ~ Variety, random = ~ar1v(Column):ar1(Row), residual =
    ~idv(units), data = nin89)
```

This model is equivalent to **2b** but with the spatial model defined as a G structure rather than an R structure. As we discussed in **1a**,

- when the G structure term involves more than one model, all but one of the models must be a correlation model (Section 4.5). In this example ar1v() is the variance model.

Modelling `Column:Row` as a G structure is a useful approach to handling incomplete arrays because not all combinations of the levels of `Row` and the levels of `Column` need to be present in the data.

Table 4.1: Sequence of variance structures for the NIN field trial

	asreml() call	random term (G)	model		residual error term (R)	model	
			1	2		1	2
1	yield ~ Replicate + Variety	-	-	-	units	idv()	-
1a	yield ~ Variety, random = ~ idv(Replicate), residual = ~ idv(units)	Replicate	idv()	-	units	idv()	-
2	yield ~ Variety, residual = ~ idv(Column):ar1(Row)	-	-	-	Column.Row	idv()	ar1()
2a	yield ~ Variety, residual = ~ ar1v(Column):ar1(Row)	-	-	-	Column.Row	ar1v()	ar1()
2b	yield ~ Variety, random = ~ idv(units), residual = ~ ar1v(Column):ar1(Row)	units	idv()	-	Column.Row	ar1v()	ar1()
3	yield ~ Variety, random = ~ ar1v(Column):ar1(Row) residual = ~ idv(units)	Column.Row	ar1v()	ar1()	units	idv()	-

4.1.1 An alternative fitting algorithm based on the gamma parameterization

In all of the RCB data models above, the residual specifies a variance model with a single variance parameter and an alternative fitting algorithm can avoid specification of an initial residual variance and lead to speedier convergence. This can easily be actioned by setting `asreml.options(gammaPar=TRUE)` before the call to asreml(). Note that `asreml.options(gammaPar=TRUE)` will set the `gammaPar` to `TRUE` for the duration of the session unless it is reset to `FALSE`. Section 4.1.2 discusses various more succinct default options. Section 4.8 provides a series of simple RCB data examples in this gamma parameterization with some alternative more implicit default specifications, and comments on the summaries.

4.1.2 Reducing the specification using defaults

Some of the specification can be reduced by using defaults. In asreml() the scaled independent variance structure (`idv(units)`, $\boldsymbol{R} = \sigma_e^2 \boldsymbol{I}_{224}$ for an RCB model for the NIN data) is the default for error. This simple error term is implicit in the model and it is not necessary to formally specify it with the residual argument. If a term in a random variance model structure is specified without a variance model function, the effects are assumed to be independent and identity functions and a residual variance parameter are added to ensure the full term is a variance structure with the variance term labelled in the output using the components of the linear model. For example, A and id(A) become σ_e^2id(A) and the variance term is labelled A. The functions A:B, A:id(B) and id(A):B all become σ_e^2id(A):id(B) and the function A:ar1(B) becomes σ_e^2id(A):ar1(B). In these 4 cases the variance is labelled A:B. Note that in cases when the residual variance model is not specified or is not completely specified and implies a scaled identity matrix, the gamma parameterization is used. So

```
> rcb.asr <- asreml(yield ~ Variety, random = ~Replicate, data = nin89)
```

and

```
> rcb.asr <- asreml(yield ~ Variety, random = ~Replicate, residual = ~units, data = nin89)
```

are equivalent to

```
> rcb.asr <- asreml(yield ~ Variety, random = ~idv(Replicate), residual = ~id(units), data
    = nin89)
```

and

```
> asreml.options(gammaPar = TRUE)
> rcb.asr <- asreml(yield ~ Variety, random = ~idv(Replicate), residual = ~idv(units), data
    = nin89)
```

4.2 Types of variance models

Three types of variance model are used in fitting R and G structures in ASReml-R, namely, *correlation models*, *homogeneous variance models* and *heterogeneous variance models*. These determine the form for each component of G and R. In the following, we denote the variance matrix of any component relating to a term in random or residual by $\boldsymbol{\Sigma}$.

4.2.1 Correlation models

In correlation models all diagonal elements are identically equal to 1. Algebraically, if $\boldsymbol{\Sigma} = [\rho_{ij}]$, $i, j = 1 \ldots \omega$, denotes the correlation matrix for a particular model, then

$$\boldsymbol{\Sigma} = [\rho_{ij}] \quad : \quad \begin{cases} \rho_{ii} = 1, & \forall i \\ \rho_{ij} = \rho_{ji}, & |\rho_{ij}| \leq 1,\ i \neq j. \end{cases}$$

The simplest correlation model in ASReml-R is the id() model, where $\boldsymbol{\Sigma} = \boldsymbol{I}_\omega$

4.2.2 Homogeneous variance models

If the variance model is specified as a homogeneous variance model, the diagonal elements all have the same positive value, σ^2 say. That is,

$$\boldsymbol{\Sigma} = [\sigma_{ij}] \quad : \quad \begin{cases} \sigma_{ii} = \sigma^2, & \forall i \\ \sigma_{ij} = \sigma_{ji}, & i \neq j. \end{cases}$$

Note that if $\boldsymbol{\Sigma}$ is a correlation model, a homogeneous variance model (with one extra parameter) is formed as $(\sigma^2 \boldsymbol{I})\boldsymbol{\Sigma}$.

For example, the homogeneous variance model corresponding to id() is idv() where $\boldsymbol{\Sigma} = \sigma^2 \boldsymbol{I}_\omega$ (or $\boldsymbol{\Sigma} = \gamma \boldsymbol{I}_\omega$).

4.2.3 Heterogeneous variance models

The third variance model is the *heterogeneous* variance model in which the diagonal elements are positive but differ. That is,

$$\boldsymbol{\Sigma} = [\sigma_{ij}] \quad : \quad \begin{cases} \sigma_{ii} = \sigma_i^2, & i = 1 \ldots \omega \\ \sigma_{ij} = \sigma_{ji}, & i \neq j. \end{cases}$$

For the models defined in terms of correlation matrices, allowance for unequal variances can be made by applying a diagonal matrix $\boldsymbol{D}$ of standard errors to the correlation matrix to generate a heterogeneous variance model. That is $\boldsymbol{D}^{1/2}\boldsymbol{\Sigma}\boldsymbol{D}^{1/2}$ In this case, ω extra parameters are added to the vector of initial values.

For example, the heterogeneous variance model corresponding to id() is idh() where $\boldsymbol{\Sigma} = \text{diag}(\sigma_1, \ldots, \sigma_\omega)$.

4.2.4 Positive definite matrices

Formation of the mixed model equations (MME) requires the inversion of the variance matrix in the R and G structures. We therefore normally require these matrices to be non-singular. The 2 exceptions are the fa model which has been specifically designed to fit singular matrices (Thompson et al.; 2003) and when singular known relationship matrices are used when ASReml-R takes account of the implied constraints in the singular relationship matrices.

4.3 Variance model functions

ASReml-R has a wide range of variance models that can be used to specify the variance matrix of terms in the random and residual formulae. The following considers the various models in terms of functional groups and describes their syntax and application.

In general, the correlation models described in the following sections have corresponding variance models whose names are simply derived by appending "v" or "h" to the correlation function name. In the former case this yields a homogeneous variance model while the latter gives the corresponding

heterogeneous model. For example, for the simple correlation model cor(), there also exists the variance functions corv() and corh(). The existence or otherwise of such models is noted for each functional group in the section detailing initial model parameter values.

4.3.1 Default identity

id(obj)
idv(obj, init=NA)
idh(obj, init=NA)

Required arguments

obj a factor in the data frame.

Optional arguments

init a vector of initial parameter values. This vector can have an optional names attribute to set the boundary constraint for each parameter. In this case, the name of each element may be one of "P", "U" or "F" for positive, unconstrained or fixed, respectively.

model		number of parameters		
f	form:	f()	fv()	fh()
id		0	1	n

Details

ASReml-R uses the id() correlation model or the idv() simple variance component model, depending on context (see the rules for combining variance models in Section 4.5) for terms in the random or residual formulae that have no variance model explicitly specified.

4.3.2 Time series type models

ar1(obj, init=NA)
ar2(obj, init=NA)
ar3(obj, init=NA)
sar(obj, init=NA)
sar2(obj, init=NA)
ma1(obj, init=NA)
ma2(obj, init=NA)
arma(obj, init=NA)

Description

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).

Required arguments

obj a factor in the data frame.

Optional arguments

init a vector of initial parameter values. This vector can have an optional names attribute to set the boundary constraint for each parameter. In this case, the name of each element may be one of "P", "U" or "F" for positive, unconstrained or fixed, respectively.

model (**f**)	form:	number of parameters **f**()	**fv**()	**fh**()
ar1		1	2	$1+n$
ar2		2	3	$2+n$
ar3		3	4	$3+n$
sar		1	2	$1+n$
sar2		2	3	$2+n$
ma1		1	2	$1+n$
ma2		2	3	$2+n$
arma		2	3	$2+n$

4.3.3 Metric based models in $\Re$ or $\Re^2$

exp(x, init=NA, dist=NA)
gau(x, init=NA, dist=NA)
lvr(x, init=NA, dist=NA)
iexp(x, y, init=NA)
igau(x, y, init=NA)
ieuc(x, y, init=NA)
sph(x, y, init=NA)
cir(x, y, init=NA)
aexp(x, y, init=NA)
agau(x, y, init=NA)
mtrn(x, y, phi=NA, nu=0.5, delta=1.0, alpha=0.0, lambda=2)

Description

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

Required arguments

x a field in the data frame containing the x coordinates. For one dimensional models,. coordinates are obtained as unique(x) or, if specified, from the component named x in the pwr.points argument to asreml().

y a field in the data frame containing the y coordinates.

Optional arguments

dist for one dimensional models, a vector of coordinates; an alternative way to specify distance information for x.

init a vector of initial parameter values. This vector can have an optional names attribute to set the boundary constraint for each parameter. In this case, the name of each element may be one of "P", "U" or "F" for positive, unconstrained or fixed, respectively.

model (f)	form:	number of parameters f()	fv()	fh()
exp		1	2	$1+n$
gau		1	2	$1+n$
lvr		1	2	$1+n$
iexp		1	2	$1+n$
igau		1	2	$1+n$
ieuc		1	2	$1+n$
sph		1	2	$1+n$
cir		1	2	$1+n$
aexp		2	3	$2+n$
agau		2	3	$2+n$

phi the range parameter. Default: $\phi =$ NA.
nu the smoothness parameter. Default: $\nu = 0.5$.
delta governs geometric anisotropy. Default: $\delta = 1.0$.
alpha governs geometric anisotropy. Default: $\alpha = 0.0$.
lambda specifies the choice of metric. Default: $\lambda = 2$ for Euclidean distance.

For the Matérn function, if an argument 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.

Details

Kriging models apply to points in an irregular (or regular) spatial grid. They require the specification of the data coordinates to calculate pairwise distances. For example,

- the distance between time points in a one-dimensional longitudinal analysis
- the spatial distance between plot coordinates in a two-dimensional field trial analysis.

Distance information for power models is obtained from the object(s) or arguments passed to the relevant special function.

For **one-dimensional** models, the distances are obtained from one of:

1. unique(*x*) where *x* is the required argument to the model function identifying the field in the data frame containing the points.

2. the dist argument to the model function.

3. the pwr.points list argument to asreml().

For **two-dimensional** models, the special functions require two arguments nominating fields in the data frame specifying the (x, y) coordinates of each observation. For example, in the analysis of spatial data, if the x coordinate was in a variate row and the y coordinate was in a variate labelled column, an anisotropic exponential model could be fitted by aexp(row, column).

Note that for an R structure the data order is assumed correct, for example, the data must be ordered *rows within ranges* for a separable autoregressive spatial model of order 1 specified as `ar1(Range):ar1(Row)`, otherwise an error is generated.

The Matérn class

ASReml-R uses an extended Matérn class which accommodates geometric anisotropy and a choice of metrics for random fields observed in two dimensions. This extension, described in detail in Haskard (2006), is given by

$$\rho(\boldsymbol{h};\boldsymbol{\phi}) = \rho_M(d(\boldsymbol{h};\delta,\alpha,\lambda);\phi,\nu)$$

where $\boldsymbol{h} = (h_x, h_y)^T$ is the spatial separation vector, (δ, α) governs geometric anisotropy, (λ) specifies the choice of metric and (ϕ, ν) are the parameters of the Matérn correlation function. The function is

$$\rho_M(d;\phi,\nu) = \left\{2^{\nu-1}\Gamma(\nu)\right\}^{-1}\left(\frac{d}{\phi}\right)^{\nu} K_\nu\left(\frac{d}{\phi}\right), \tag{4.1}$$

where $\phi > 0$ is a range parameter, $\nu > 0$ is a smoothness parameter, $\Gamma(\cdot)$ is the gamma function, $K_\nu(.)$ is the modified Bessel function of the third kind of order ν (Abramowitz and Stegun, 1965, section 9.6) and d is the distance defined in terms of X and Y axes: $h_x = x_i - x_j$; $h_y = y_i - y_j$; $s_x = \cos(\alpha)h_x + \sin(\alpha)h_y$; $s_y = \cos(\alpha)h_x - \sin(\alpha)h_y$; $d = (\delta|s_x|^\lambda + |s_y|^\lambda/\delta)^{1/\lambda}$.

For a given ν, the range parameter ϕ affects the rate of decay of $\rho(\cdot)$ with increasing d. The parameter $\nu > 0$ controls the analytic smoothness of the underlying process $\boldsymbol{u}_s$, the process being $\lceil\nu\rceil - 1$ times mean-square differentiable, where $\lceil\nu\rceil$ is the smallest integer greater than or equal to ν (Stein, 1999, page 31). Larger ν correspond to smoother processes. ASReml-R uses numerical derivatives for ν when its current value is outside the interval [0.2,5].

When $\nu = m + \frac{1}{2}$ with m a non-negative integer, $\rho_M(\cdot)$ is the product of $\exp(-d/\phi)$ and a polynomial of degree m in d. Thus $\nu = \frac{1}{2}$ yields the exponential correlation function, $\rho_M(d;\phi,\frac{1}{2}) = \exp(-d/\phi)$, and $\nu = 1$ yields Whittle's elementary correlation function, $\rho_M(d;\phi,1) = (d/\phi)K_1(d/\phi)$ (Webster and Oliver, 2001).

When $\nu = 1.5$ then

$$\rho_M(d;\phi,1.5) = \exp(-d/\phi)(1 + d/\phi)$$

which is the correlation function of a random field which is continuous and once differentiable. This has been used recently by Kammann and Wand (2003). As $\nu \to \infty$ then $\rho_M(\cdot)$ tends to the gaussian correlation function.

The metric parameter λ is not estimated by ASReml-R; it is usually set to 2 for Euclidean distance. Setting $\lambda = 1$ provides the cityblock metric, which together with $\nu = 0.5$ models a separable AR1×AR1 process. Cityblock metric may be appropriate when the dominant spatial processes are aligned with rows/columns as occurs in field experiments. Geometric anisotropy is discussed in most geostatistical books (Webster and Oliver; 2001; Diggle et al.; 2003) but rarely are the anisotropy angle or ratio estimated from the data. Similarly the smoothness parameter ν is often

set a-priori (Kammann and Wand; 2003; Diggle et al.; 2003). However Stein (1999) and Haskard et al. (2007) demonstrate that ν can be reliably estimated even for modest sized data-sets, subject to caveats regarding the sampling design.

Estimation

The order of the parameters in mtrn(), with their defaults, is (ϕ, $\nu = 0.5$, $\delta = 1$, $\alpha = 0$, $\lambda = 2$). Parameters are fixed or estimated depending on the data type (numeric or character) of the argument to the respective parameter.

- If an argument 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.

For example, to fit a Matérn model with only ϕ estimated and the other parameters set at their defaults then we could use mtrn(phi = 0.1) where the starting value for estimation is given as 0.1.

To fix ν some value other than the default and estimate ϕ, the fixed value and constraint code are given as a single string to the nu argument. That is mtrn(phi = 0.1, nu = "1.0F")

The parameters ϕ and ν are highly correlated so it may be better to manually cover a grid of ν values.

We note that there is non-uniqueness in the anisotropy parameters of this metric $d(\cdot)$ since inverting δ and adding $\frac{\pi}{2}$ to α gives the same distance. This non-uniqueness can be removed by constraining $0 \leq \alpha < \frac{\pi}{2}$ and $\delta > 0$, or by constraining $0 \leq \alpha < \pi$ and either $0 < \delta \leq 1$ or $\delta \geq 1$. With $\lambda = 2$, isotropy occurs when $\delta = 1$, and then the rotation angle α is irrelevant: correlation contours are circles, compared with ellipses in general. With $\lambda = 1$, correlation contours are diamonds.

4.3.4 General structure models

cor(obj, init=NA)
corb(obj, k=1, init=NA)
corg(obj, init=NA)
diag(obj, init=NA)
us(obj, init=NA)
chol(obj, k=1, init=NA)
cholc(obj, k=1, init=NA)
ante(obj, k=1, init=NA)
sfa(obj, k=1, init=NA)
fa(obj, k=1, init=NA)
facv(obj, k=1, init=NA)
rr(obj, k=1, init=NA)

Description

4.3 Variance model functions

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 structure (fa).

Required arguments

obj a factor in the data frame.

Optional arguments

init a vector of initial parameter values. This vector can have an optional names attribute to set the boundary constraint for each parameter. In this case, the name of each element may be one of "P", "U" or "F" for positive, unconstrained or fixed, respectively.

model (**f**)	form:	number of parameters **f**()	**fv**()	**fh**()
cor		1	2	$1+n$
corb		k	$k+1$	$k+n$
corg		$n(n-1)/2$	$1+n(n-1)/2$	$n+n(n-1)/2$
diag		n		
us		$n(n+1)/2$		
chol[1]		$n(n+1)/2$		
cholc[1]		$n(n+1)/2$		
ante[1]		$n(n+1)/2$		
sfa		$kn+n$		
fa		$kn+n$		
facv		$kn+n$		
rr		kn		

[1] chol, cholc and ante models have $(k+1)(n-k/2)$ parameters but $n(n+1)/2$ initial values row-wise from the lower triangle of an unstructured matrix are given and converted to the appropriate parameterization.

k the number of subdiagonal bands for corb
the order of the Cholesky decomposition for chol and cholc
the order of antedependence (ante) and factor analytic models (fa).

Details

The k-factor Cholesky structure models $\boldsymbol{\Sigma}^{\omega\times\omega}$ as

$$\boldsymbol{\Sigma} = \boldsymbol{L}\boldsymbol{D}\boldsymbol{L}'$$

where $\boldsymbol{L}^{\omega\times\omega}$ is a unit lower triangular matrix and $\boldsymbol{D} = \mathrm{diag}(d_1, \ldots, d_\omega)$.

In the chol(,k) factorization $\boldsymbol{L}$ has k non-zero (unequal) bands below the diagonal, that is, the elements $\{l_{ij}\}$ of $\boldsymbol{L}$ are

$$\begin{aligned} l_{ii} &= 1 \\ l_{ij} &= v_{ij}.\ 1 \le i-j \le k \\ l_{ij} &= 0,\ otherwise \end{aligned}$$

In the chol(,k) factorization $\boldsymbol{L}$ has columns $\boldsymbol{l}_i = (l_{1i}, \ldots, l_{\omega i})'$ where

$$\begin{aligned} l_{ii} &= 1 \\ l_{ij} &= 0 \text{ for } i < j,\ k < j < i \end{aligned}$$

For example, if a factor Site has 4 levels then

```
asreml(...,cholc(Site,1)...)
```

generates $\boldsymbol{\Sigma} = \boldsymbol{LDL}'$ where

$$\boldsymbol{L} = \begin{bmatrix} 1 & & & \\ l_{21} & 1 & & \\ l_{31} & 0 & 1 & \\ l_{41} & 0 & 0 & 1 \end{bmatrix} \quad \boldsymbol{D} = \begin{bmatrix} d_1 & 0 & 0 & 0 \\ 0 & d_2 & 0 & 0 \\ 0 & 0 & d_3 & 0 \\ 0 & 0 & 0 & d_4 \end{bmatrix}$$

This form is similar to a Factor Analytic model. In ASReml-R the initial parameters for both Cholesky factorizations are given as the lower triangle row-wise of an unstructured matrix and converted internally to the appropriate factorization. So, if

$$\boldsymbol{\Sigma} = \begin{bmatrix} \sigma_{11} & & & \\ \sigma_{21} & \sigma_{22} & & \\ \sigma_{31} & \sigma_{32} & \sigma_{33} & \\ \sigma_{41} & \sigma_{42} & \sigma_{43} & \sigma_{44} \end{bmatrix}$$

the initial values are given as

c($\sigma_{11}, \sigma_{21}, \sigma_{22}, \ldots, \sigma_{44}$)

The k-factor antedependence ante(,k) structure models $\boldsymbol{\Sigma}^{\omega \times \omega}$ as

$$\boldsymbol{\Sigma}^{-1} = \boldsymbol{UDU}'$$

where $\boldsymbol{U}^{\omega \times \omega}$ is a unit upper triangular matrix with elements $\{u_{ij}\}$ where

$$\begin{aligned} u_{ii} &= 1 \\ u_{ij} &= 0, i > j \\ u_{ij} &= u_{ij}, 1 \leq i - j \leq k \end{aligned}$$

and $\boldsymbol{D} = \text{diag}(d_1, \ldots, d_\omega)$.

Considering the above example for a factor Site with 4 levels,

```
asreml(...,ante(Site,1)...)
```

generates $\boldsymbol{\Sigma}^{-1} = \boldsymbol{UDU}'$ where

$$\boldsymbol{U} = \begin{bmatrix} 1 & u_{12} & 0 & 0 \\ 0 & 1 & u_{23} & 0 \\ 0 & 0 & 1 & u_{34} \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \boldsymbol{D} = \begin{bmatrix} d_1 & 0 & 0 & 0 \\ 0 & d_2 & 0 & 0 \\ 0 & 0 & d_3 & 0 \\ 0 & 0 & 0 & d_4 \end{bmatrix}$$

4.3 Variance model functions

In ASReml-R the parameters for ante are given as for us, chol and cholc as the lower triangle row-wise of an unstructured matrix.

The functions facv(), fa() and sfa() are different parameterizations of the factor analytic model. In the first two for models of order k (facv(, k) and fa(k)) the variance matrix $\mathbf{\Sigma}^{\omega\times\omega}$ is modelled as

$$\mathbf{\Sigma} = \mathbf{\Gamma}\mathbf{\Gamma}' + \mathbf{\Psi}$$

where $\mathbf{\Gamma}^{(\omega\times k)}$ is a matrix of loadings and $\mathbf{\Psi}^{\omega\times\omega}$ is a diagonal matrix whose elements are referred to as specific variances. As the covariances are modelled by the loadings, cv is included in the function name.

For example, if Site is a factor with 4 levels, the component matrices for asreml(. . . ,facv(Site,1). . .) are

$$\mathbf{\Gamma} = \begin{bmatrix} l_1 \\ l_2 \\ l_3 \\ l_4 \end{bmatrix} \quad \mathbf{\Psi} = \begin{bmatrix} \psi_1 & 0 & 0 & 0 \\ 0 & \psi_2 & 0 & 0 \\ 0 & 0 & \psi_3 & 0 \\ 0 & 0 & 0 & \psi_4 \end{bmatrix}$$

where the parameters are given in the order c(vec($\mathbf{\Gamma}$), diagv($\mathbf{\Psi}$)), where diagv() is the operator that puts the diagonal terms of a square matrix into a vector.

Alternatively the variance-covariance matrix $\mathbf{\Sigma}^{\omega\times\omega}$ can be scaled in sfa(,k) using a correlation scale with $\mathbf{\Sigma} = \boldsymbol{DCD}$, where $\boldsymbol{D}^{\omega\times\omega}$ is diagonal such that $\boldsymbol{DD} = diag\,(\mathbf{\Sigma})$ ie. a diagonal matrix with diagonal elements the diagonal elements of $\mathbf{\Sigma}$, and $\boldsymbol{C}^{\omega\times\omega}$ is a correlation matrix of the form $\boldsymbol{FF}^{\mathsf{T}} + \boldsymbol{E}$, where $\boldsymbol{F}$ is a matrix of loadings on the correlation scale and $\boldsymbol{E}$ is diagonal and is defined by difference. Comparison of $\mathbf{\Sigma}$ under the two parameterization show that $\boldsymbol{DF} = \mathbf{\Gamma}$ and $\boldsymbol{DED} = \mathbf{\Psi}$. The parameters for the sfa(,k) model are specified in the order the loadings for each factor ($\boldsymbol{F}$) followed by the variances (the diagonal elements of $\mathbf{\Sigma}$ or $\boldsymbol{DD}$). Note that the parameters for the facv(,k) model are ordered as for the sfa(,k) model.

The third form of the factor analytic model is fa(,k) and has the same parameterisation as for facv(,k). The difference is that this model introduces the k factor effects directly into an extended linear model and in the estimation procedure. This formulation is computationally faster than the facv(, k) and sfa(, k) formulations for large problems when k is much smaller than ω, and permits some elements of $\mathbf{\Psi}$ to be fixed as zero. It also allows easier specification for prediction of factor effects. Slightly confusingly, but in the interests of upward compatibility, with fa(,k) the parameters are ordered in the reverse order, c(diagv($\mathbf{\Psi}$), vec($\mathbf{\Gamma}$)).

4.3.4.1 Limitations

The functions fa(,k) and sfa(,k), unlike facv(,k), do not allow singular $\mathbf{\Sigma}$ matrices to be estimated. Constraints are required in $\mathbf{\Gamma}$ for $k > 1$ for identifiability. 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 ($2 \le i \le k$) 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 $\mathbf{\Sigma}$.

Unfortunately because the fa(,*k*) formulation allows singular variance matrices it is not available in residual (R) structures.

4.3.4.2 Updating loadings in factor analytic models

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 user can set the initial value of p with the option ai.penalty=*p*. After the penalty has reduced to 1%, it is further reduced to 0.2%. The qualifier can be used to set p to 0 if desired. Another option, ai.loadings, 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.

4.3.5 Special case of fa(): rr()

In a reduced rank factor analytic model of order k (rr(,*k*)), the variance matrix $\mathbf{\Sigma}^{\omega\times\omega}$ is modelled as

$$\mathbf{\Sigma} = \mathbf{\Gamma}\mathbf{\Gamma}'$$

where $\mathbf{\Gamma}^{(\omega\times k)}$ is a matrix of loadings.

For example, if Site is a factor with 4 levels, the component matrix for asreml(. . . ,rr(Site,1). . .) is simply

$$\mathbf{\Gamma} = \begin{bmatrix} l_1 \\ l_2 \\ l_3 \\ l_4 \end{bmatrix}.$$

Note that for computational reasons there can only be one rr() term in a compound model term, and it is recommended that the rr() term is the first term. In a 3-way structure with an rr() term,

if a relationship structure (specified using vm() in ASReml-R Version 4) is used then it should be the third term and only an identity term can be used for the second term.

Note also that the summary() function currently returns zeros for the set of specific variances and they also appear in the table of variance parameters using start.values = TRUE. The user needs to be aware of this and to extract them as necessary when post-processing.

4.3.6 Known relationship structures

vm(obj, source, singG=NULL)

ide(obj, source)

Arguments

obj a factor in the data frame.

source The known inverse or relationship matrix:

- a sparse inverse variance matrix held in three column co-ordinate form in row major order. This triplet matrix must have class ginv from a call to ainverse(), or have attribute INVERSE set to TRUE. For backwards compatibility, a three column data frame is also accepted. In either case, the source must have a rowNames attribute.
- a sparse relationship matrix held in three column co-ordinate form (as a matrix) in row major order. If the attribute INVERSE is not set then FALSE is assumed; a rowNames attribute must be set.
- a matrix (or Matrix object) with a dimnames attribute giving the levels of the model term being defined. This may be a relationship matrix or its inverse; if an inverse, it must have an attribute INVERSE set to TRUE.
- a numeric vector of the lower triangular elements in row major order. The vector must have a rowNames attribute, and if an inverse structure, it must also have an INVERSE attribute set to TRUE.

singG This argument is ignored if source has class ginv or its INVERSE attribute is TRUE; in such cases source must be one of:

- a sparse matrix in coordinate form with class ginv, or INVERSE attribute set to TRUE.
- an object of class matrix or Matrix with INVERSE attribute set to TRUE.
- a vector assumed to be the lower triangle in row major order with INVERSE attribute set to TRUE.

If source does not have class ginv, or the attribute INVERSE is FALSE or is not set, and singG is NULL (the default), then source is assumed to be a positive definite relationship matrix and singG is reset to "PD". Otherwise, a character string giving the state of the (to be inverted) source object:

PD source is positive definite (default).

ND source is non-singular indefinite (positive and negative roots). ASReml-R ignores the indefinite

condition and proceeds.

PSD source is positive semi-definite. In this case, ASReml-R 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.

NSD source is singular indefinite (positive, zero and negative roots). The indefinite condition is ignored and ASReml-R proceeds using Lagrangian multipliers as for PSD matrices.

Details
If source inherits from class Matrix, ASReml-R will convert source internally to either sparse triplet form (class dsparseMatrix), or dense vector form (class ddenseMatrix) for processing. The names of the levels of vm() are given by the rowNames attribute associated with source. The number of levels of vm(obj, source) might be greater than the levels of obj in the data frame. ide(obj, source) creates a term with the levels associated with source, and modelled by the homogeneous form of the identity variance structure. If an ide() term succeeds its partner vm() term then source can be omitted from ide() and ASReml-R uses the source from the partner vm().

Warning: If a model term is specified in terms of vm() and other model terms then, at present, the other model terms must specify a variance, as opposed to correlation, matrix.

4.3.6.1 Linking a relationship matrix to 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 $\boldsymbol{Z}\boldsymbol{u}$ ($\boldsymbol{Z}$ is the design matrix linking observations to factor levels) and a variance structure for $\boldsymbol{u}$ based on $\boldsymbol{K} = \boldsymbol{M}\boldsymbol{M}'$, 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 `asreml` 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.

4.3.7 General variance structures

str(form, vmodel)

Required arguments

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

vmodel a formula object containing asreml variance functions separated by ":" operators specifying the direct product structure that applies to the set of terms in form. The size of the variance structures can be given as an integer argument to the variance functions in place of the usual *factor* object.

Details

Typically a variance structure applies to an individual term in the linear model, with no covariance between model terms. Sometimes it is appropriate, for example in random regression models, to include a covariance parameter. The model terms in form are kept together and identified by the first term in the sequence. The variance structure defined in vmodel begins at the first term and covers the subsequent terms in the sequence. The overall size of the variance model is checked against the total number of levels of the terms in form, however, the sequence of effects matching the variance structure definition is not checked.

For example, in the first order random coefficient regression model it is required to specify a covariance between the intercept and slope for each subject to ensure translation invariance, that is, equivalent variance parameter estimates for addition of any constant to the independent variable. For example, in a random coefficient regression where a set of random intercepts is specified by the model term `Animal` (with 10 levels) and a set of random slopes is specified by the model term `age.Animal`, translation invariance is achieved using str() as

```
str(Animal + age.Animal, us(2):id(10))
```

The algorithm places the model terms specified using the argument form together in the processed random model, here `Animal` followed by `age.Animal`. The variance structure(s) begins at the start of the first term specified in str() and is expected to exactly span the whole set of terms given within the brackets. The overall size of the variance model is checked against the total number of levels of these terms, but the user must verify that the ordering is appropriate for (matches) the variance model specified.

In our example, this random model generates a combined set of random effects from the individual animal intercepts, $\boldsymbol{u}_I = (u_{I1} \dots u_{I10})^\mathsf{T}$ and animal slopes, $\boldsymbol{u}_S = (u_{S1} \dots u_{S10})^\mathsf{T}$, as $\boldsymbol{u}_{IS} = (\boldsymbol{u}_I^\mathsf{T}\, \boldsymbol{u}_S^\mathsf{T})^\mathsf{T}$. This term then has variance structure of the form

$$\text{var}\,(\boldsymbol{u}_{IS}) = \text{var}\left(\begin{bmatrix} \boldsymbol{u}_I \\ \boldsymbol{u}_S \end{bmatrix}\right) = \begin{bmatrix} \sigma_{II} & \sigma_{IS} \\ \sigma_{IS} & \sigma_{SS} \end{bmatrix} \otimes \boldsymbol{I}_{10} = \begin{bmatrix} \sigma_{II}\boldsymbol{I}_{10} & \sigma_{IS}\boldsymbol{I}_{10} \\ \sigma_{IS}\boldsymbol{I}_{10} & \sigma_{SS}\boldsymbol{I}_{10} \end{bmatrix}$$

Here, the set of animal intercepts has a common variance (σ_{II}), and the set of animal slopes has a (different) common variance (σ_{SS}). Intercepts and/or slopes from two different animals are independent, but the intercept and slope from any given animal have covariance σ_{IS} (or correlation $\sigma_{IS}/\sqrt{\sigma_{II}\sigma_{SS}}$). In this context, we use integers as arguments to emphasize that the arguments are specifying the size of the variance structure. For this example, `id(10)` can be replaced by `id(Animal)`. This random regression model has been developed to describe the form of the str() function. We note that this model is equivalent to

```
us(pol(age)).id(Animal)
```

Note that model terms with variance functions such as fa(), rr(), vm() and ide() that generate new model factors with a modified set of levels must be given explicitly in the form argument. This ensures that the overall sizes (in terms of the total numbers of levels) of form and vmodel conform and ensures the correct identification of terms in the model, especially in predict statements.

An example is from the use of reduced animal models with the need to form a composite design matrix as the sum of half a sire and half a dam matrix. We first set up in the data frame a variate with values 0.5.

```
data.df$half <- rep(0.5, nrow(data.df))
```

and then form a design matrix from sires (`P.Male`) and dams (`P.Female`) scaled by a factor 0.5 using the variate `half`:

```
str(~fa(YrLoc, 1):vm(P.Male, Ainv):half + and(fa(YrLoc, 1):vm(P.Female, Ainv):half),
    ~fa(YrLoc, 1):vm(P.Male, Ainv))
```

Note that the `YrLoc` effects are extended by 1 to include the one extra factor in `fa(,1)`, and the levels associated with `P.Male` and `P.Female` are extended to include all levels in `Ainv`. We note in passing that the functionality of and() allows

```
and(fa(YrLoc, 1):vm(P.Female, Ainv):half)
```

to be written in the alternative form

```
and(fa(YrLoc, 1):vm(P.Female, Ainv), 0.5)
```

4.4 Default initial values for variance parameters

The default initial values are 0.1 for both variance ratios and correlations, and 0.1*v for variance components, where v is half the simple variance of the response. The corresponding default parameter constraints are P (positive) for variance ratios, U (unconstrained) for correlations and P for variance components. These defaults can be altered using the methods described in Section 3.7.1.

4.5 Rules for combining variance models

Variance structures are sometimes formed by combining variance models. For example, a two factor interaction may involve two variance models, one for each of the two factors in the interaction. Some of the rules for combining variance models differ for R structures and G structures. The following rules apply:

- when combining variance models in both R and G structures, the resulting direct product structure must match the ordered effects with the outer factor first. For example, the NIN data are ordered rows within columns. This is why in **Model 3** (page 47) the ar1v() variance model for Column is specified first in the interaction term.
- ASReml-R automatically includes and estimates an error variance parameter for each section of an R structure that is a correlation matrix.
- when the G structure involves more than one variance model, one must be either an homogeneous or a heterogeneous variance model and the rest should be correlation models; if more than one are non-correlation models then constraints should be used to avoid identifiability problems, that is, to prevent attempts to estimate confounded parameters.

4.6 Constraining variance parameters

4.6.1 The vcc argument to asreml()

vcc is the argument to asreml() that allows users the functionality of the `vcm` 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

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

4.6.2 The vcm argument to asreml()

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

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

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

$$\begin{array}{ll} \texttt{Trial\_1!R} & \left[\begin{array}{ccc} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{array}\right] \\ \texttt{Trial\_1!Row!cor} & \\ \texttt{Trial\_1!Column!cor} & \\ \texttt{Trial\_2!R} & \\ \texttt{Trial\_2!Row!cor} & \\ \texttt{Trial\_2!Column!cor} & \end{array}$$

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.

4.7 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("../examples/coop.dat", header = TRUE)
> head(coop)
```

```
     tag Sire Dam Grp Sex Brr Litter age  wwt  ywt gfw fdm fat
1 500001    1   1  18   2   2   2737  31 37.0 48.0 3.2  NA  NA
2 500002    1   2  18   2   3   2738  40 28.5 42.5 2.7  NA  NA
3 500003    1   2  18   1   3   2738  40 30.0 49.0 2.5  NA  NA
4 500004    1   3  18   1   2   2739  46 44.0 53.5 3.0  NA  NA
5 500005    1   4  18   1   1   2740  54 43.0 59.5 2.5  NA  NA
6 500006    1   5  18   1   1   2741  34 39.5 52.5 3.5  NA  NA
```

```
> 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
```

```
                 Component Value Constraint
1  trait:Sire!trait_ywt:ywt  1.40          P
2  trait:Sire!trait_fat:ywt  0.13          P
3  trait:Sire!trait_fat:fat  0.03          P
4            units:trait!R  1.00          F
5 units:trait!trait_ywt:ywt 23.00          P
6 units:trait!trait_fat:ywt  2.50          P
7 units:trait!trait_fat:fat  1.60          P
```

```
> 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)
```

```
Model fitted using the sigma parameterization.
ASReml 4.1.0 Mon Aug 12 10:33:45 2019
         LogLik        Sigma2     DF     wall    cpu
 1     -9991.549           1.0   6164 10:33:46    0.0
 2     -9990.785           1.0   6164 10:33:46    0.0
```

```
 3     -9990.226           1.0   6164 10:33:46    0.0
 4     -9990.084           1.0   6164 10:33:46    0.0
 5     -9990.084           1.0   6164 10:33:46    0.0
```

Terms with zero df listed in attribute 'zerodf' of the wald table.

```
> 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
                      "P"                       "P"                       "P"
            units:trait!R units:trait!trait_ywt:ywt units:trait!trait_fat:ywt
                      "F"                       "P"                       "P"
units:trait!trait_fat:fat
                      "P"
```

```
> 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
```

```
                                          names constraint number
trait:Sire!trait_ywt:ywt   trait:Sire!trait_ywt:ywt          P      1
trait:Sire!trait_fat:ywt   trait:Sire!trait_fat:ywt          P      2
trait:Sire!trait_fat:fat   trait:Sire!trait_fat:fat          P      3
units:trait!R                           units:trait!R        F      4
units:trait!trait_ywt:ywt units:trait!trait_ywt:ywt          P      5
units:trait!trait_fat:ywt units:trait!trait_fat:ywt          P      6
units:trait!trait_fat:fat units:trait!trait_fat:fat          P      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
```

4.8 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 2.1.1), or the gamma parameterization (with gamma parameters, Section 2.1.6). Specifying the residual model as a variance structure (or with `dsum` for multi-section models, Section 3.8) 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.dat <- asreml.read.table("../examples/rcbdat.csv", header = TRUE, sep = ",")
> head(rcb.dat)
```

```
  Variety Site Row Range Block   yield
1    Var3  RCB   1     1     1 4420.84
2   Var44  RCB   2     1     1 4070.84
3   Var10  RCB   3     1     1 3917.50
4   Var37  RCB   4     1     1 3223.34
5   Var36  RCB   5     1     1 4028.34
6   Var35  RCB   6     1     1 3099.17
```

```
> rcb.dat$yield <- rcb.dat$yield/1000
> 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 Mon Aug 12 10:33:47 2019
```

```
          LogLik        Sigma2     DF     wall    cpu
 1      -28.7733           1.0     96 10:33:47    0.0
 2        2.7153           1.0     96 10:33:47    0.0
 3       35.8161           1.0     96 10:33:47    0.0
 4       55.9947           1.0     96 10:33:47    0.0
 5       64.0512           1.0     96 10:33:47    0.0
 6       65.0829           1.0     96 10:33:47    0.0
 7       65.1126           1.0     96 10:33:47    0.0
 8       65.1127           1.0     96 10:33:47    0.0
```

```
> summary(rcb.asr)$varcomp
```

```
             component   std.error   z.ratio bound %ch
Block!Block 0.06715427 0.068195046 0.9847383     P   0
units!units 0.05014295 0.007314089 6.8556664     P   0
units!R     1.00000000          NA        NA     F   0
```

Note that in this case $\text{var}(\boldsymbol{e}) = \sigma_e^2 \boldsymbol{I}_{144}$ and $\text{var}(\boldsymbol{u}) = \gamma_r \sigma_e^2 \boldsymbol{I}_3$ with $\sigma_r^2 = \gamma_r \sigma_e^2$. 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.

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 Mon Aug 12 10:33:49 2019
          LogLik        Sigma2     DF     wall    cpu
 1       58.2386     0.0608567     96 10:33:49    0.0
 2       60.4027     0.0578118     96 10:33:49    0.0
 3       62.6539     0.0546469     96 10:33:49    0.0
 4       64.1290     0.0524343     96 10:33:49    0.0
 5       64.8815     0.0510398     96 10:33:49    0.0
 6       65.0836     0.0504189     96 10:33:49    0.0
 7       65.1117     0.0501896     96 10:33:49    0.0
 8       65.1127     0.0501448     96 10:33:49    0.0
```

```
Model fitted using the gamma parameterization.
ASReml 4.1.0 Fri Feb 2 09:50:17 2018
LogLik Sigma2 DF wall cpu
1 58.2386 0.0608567 96 09:50:17 0.0
2 60.4027 0.0578118 96 09:50:17 0.0
```

```
3 62.6539 0.0546469 96 09:50:17 0.0
4 64.1290 0.0524343 96 09:50:17 0.0
5 64.8815 0.0510398 96 09:50:17 0.0
6 65.0836 0.0504189 96 09:50:17 0.0
7 65.1117 0.0501896 96 09:50:17 0.0
8 65.1127 0.0501448 96 09:50:17 0.0
```

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

The variance parameters reported

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 0.068019908 0.9873075     P 0.2
units!units 0.05014483          NA        NA     F 0.0
units!R     0.05014483 0.007314511 6.8555266     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 0.068019908 0.9873075     P 0.2
units!units 1.000000 0.05014483          NA        NA     F 0.0
units!R     1.000000 0.05014483 0.007314511 6.8555266     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() 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:

4.8 Switching between the gamma and sigma parameterization

```
> 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 Mon Aug 12 10:33:51 2019
          LogLik        Sigma2     DF     wall    cpu
 1       58.2386     0.0608567     96 10:33:51    0.0
 2       60.4027     0.0578118     96 10:33:51    0.0
 3       62.6539     0.0546469     96 10:33:51    0.0
 4       64.1290     0.0524343     96 10:33:51    0.0
 5       64.8815     0.0510398     96 10:33:51    0.0
 6       65.0836     0.0504189     96 10:33:51    0.0
 7       65.1117     0.0501896     96 10:33:51    0.0
 8       65.1127     0.0501448     96 10:33:51    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 3.8 for more detail.

5 Specifying variance structures using genetic and other relationships

5.1 Introduction

In many situations there are relationships between individuals and we expect the variance matrix of the individual effects to be proportional to the relationship matrix. The relationship matrix may arise from spatial, temporal, genetic or other relationships and the previous chapter has discussed models motivated by spatial and temporal relationships. This chapter describes how to specify models with general relationship or inverse relationship matrices but in addition describes methods to efficiently analyse some common genetic relationship matrices that depend on pedigree relationships. Version 3 considered two cases, generating an inverse relationship matrix from a pedigree and using the ped() function to specify the model and inputing an inverse relationship matrix using the giv() function. Version 4 unifies this by having a variance model function vm() which allows the use of either a relationship or inverse relationship matrix, input as either a matrix, a vector or as a sparse matrix or generated from a pedigree (using ainverse()). The relationship matrix is usually positive definite but ASReml-R copes with semi positive definite and singular and non-singular indefinite matrices.

In a genetic analysis we have phenotypic data on a set of individuals that are genetically linked via a pedigree. The genetic effects are therefore correlated and, assuming normal modes of inheritance, the correlation expected from additive genetic effects can be derived from the pedigree provided all the genetic links are present. The additive genetic relationship matrix (sometimes called the numerator relationship matrix, or $\boldsymbol{A}$ matrix) can be calculated from the pedigree. It is actually the ***inverse*** relationship matrix that is required by asreml() for analysis. Fortunately the inverse matrix is easier to calculate than the relationship matrix and ASReml-R takes account of this.

The inclusion of an $\boldsymbol{A}^{-1}$ matrix in an analysis is essentially a two step process:

1. the function ainverse() takes a pedigree data frame and returns the $\boldsymbol{A}^{-1}$ matrix in sparse form

2. the matrix from step 1 is included in an asreml() analysis using the vm() function.

For the more general situation, where the pedigree based inverse relationship matrix generated by ainverse() is not appropriate, the user can include a general inverse variance matrix provided its structure adheres to one of the allowable forms given in Section 5.3.

In this chapter we illustrate the procedure using the data in Harvey (1977) described in Section 1.3.4.

5.2 Pedigree and genetic groups

5.2.1 Pedigree objects

The pedigree defines the genetic relationships among individuals when fitting a genetic model. The pedigree object is simply a data frame with the following properties:

- three columns: the identity of the individual, its male parent and its female parent (or maternal grand sire if the MGS option to ainverse() is to be specified)
- is sorted so that the row giving the pedigree of an individual appears before any row where that individual appears as a parent
- uses identity 0 or NA for unknown parents.

For example, the first 20 lines of `harvey.ped` are:

```
harvey.ped <- read.table("harvey.ped", header = T, as.is = T)

head(harvey.ped, 20)
```

```
   Calf   Sire Dam
1   101 Sire_1   0
2   102 Sire_1   0
3   103 Sire_1   0
4   104 Sire_1   0
5   105 Sire_1   0
6   106 Sire_1   0
7   107 Sire_1   0
8   108 Sire_1   0
9   109 Sire_2   0
10  110 Sire_2   0
11  111 Sire_2   0
12  112 Sire_2   0
13  113 Sire_2   0
14  114 Sire_2   0
15  115 Sire_2   0
16  116 Sire_2   0
17  117 Sire_3   0
18  118 Sire_3   0
19  119 Sire_3   0
20  120 Sire_3   0
```

5.2.2 Genetic groups

If all individuals belong to one genetic group then, as above, use 0 as the identity of the parents of base individuals. However, if base individuals belong to various genetic groups, this can be specified

using the groups argument to ainverse(). The pedigree data frame must then identify these groups. All base individuals should have group identifiers as parents. In this case the identity 0 will only appear on the group identity rows, as in the following example where three sire lines are fitted as genetic groups.

```
harveyG.ped <- read.table("harveyg.ped", header = T, as.is = T)

head(harveyG.ped, 20)
```

```
     Calf   Sire Dam
1      G1      0   0
2      G2      0   0
3      G3      0   0
4  Sire_1     G1  G1
5  Sire_2     G1  G1
6  Sire_3     G1  G1
7  Sire_4     G2  G2
8  Sire_5     G2  G2
9  Sire_6     G3  G3
10 Sire_7     G3  G3
11 Sire_8     G3  G3
12 Sire_9     G3  G3
13    101 Sire_1  G1
14    102 Sire_1  G1
15    103 Sire_1  G1
16    104 Sire_1  G1
17    105 Sire_1  G1
18    106 Sire_1  G1
19    107 Sire_1  G1
20    108 Sire_1  G1
```

It is usually appropriate to allocate a genetic group identifier where the parent is unknown.

5.3 Specifying relationship and inverse relationship matrices

A symmetric matrix or symmetric inverse matrix from an external source can be included in the analysis as:

- a sparse inverse variance matrix held in three column co-ordinate form in row major order. This triplet matrix must have class ginv from a call to ainverse, or have attribute INVERSE set to TRUE. For backwards compatibility, a three column data frame is assumed to be a sparse inverse in co-ordinate form. In either case, the source must have a rowNames attribute.

- a sparse relationship matrix held in three column co-ordinate form in row major order. If the attribute INVERSE is not set then FALSE is assumed; a rowNames attribute must be set.

- a matrix (or Matrix object) with a dimnames attribute giving the levels of the model term being defined. This may be a relationship matrix or its inverse; if an inverse, it must have an attribute INVERSE set to TRUE.

- a numeric vector of the lower triangular elements in row major order. The vector must have

a rowNames attribute, and if an inverse structure, it must also have an INVERSE attribute set to TRUE.

In all cases the matrix object must have an attribute rowNames, a character vector that uniquely identifies each row of the matrix. This vector of identifiers may be a super set of the vector of levels of the corresponding factor in the data, but must at least contain all the individuals in the data.

An inverse relationship matrix can be obtained from the harvey pedigree using:

```
harvey.ainv <- ainverse(harvey.ped)
attr(harvey.ainv, "rowNames")
```

```
 [1] "Sire_1" "Sire_2" "Sire_3" "Sire_4" "Sire_5" "Sire_6" "Sire_7" "Sire_8" "Sire_9"
[10] "101"    "102"    "103"    "104"    "105"    "106"    "107"    "108"    "109"
[19] "110"    "111"    "112"    "113"    "114"    "115"    "116"    "117"    "118"
[28] "119"    "120"    "121"    "122"    "123"    "124"    "125"    "126"    "127"
[37] "128"    "129"    "130"    "131"    "132"    "133"    "134"    "135"    "136"
[46] "137"    "138"    "139"    "140"    "141"    "142"    "143"    "144"    "145"
[55] "146"    "147"    "148"    "149"    "150"    "151"    "152"    "153"    "154"
[64] "155"    "156"    "157"    "158"    "159"    "160"    "161"    "162"    "163"
[73] "164"    "165"
```

```
harvey.ainv[1:20, ]
```

```
      Row Column   Ainverse
 [1,]   1      1  3.6666667
 [2,]   2      2  3.6666667
 [3,]   3      3  2.6666667
 [4,]   4      4  3.6666667
 [5,]   5      5  3.3333333
 [6,]   6      6  3.0000000
 [7,]   7      7  3.6666667
 [8,]   8      8  3.3333333
 [9,]   9      9  3.6666667
[10,]  10      1 -0.6666667
[11,]  10     10  1.3333333
[12,]  11      1 -0.6666667
[13,]  11     11  1.3333333
[14,]  12      1 -0.6666667
[15,]  12     12  1.3333333
[16,]  13      1 -0.6666667
[17,]  13     13  1.3333333
[18,]  14      1 -0.6666667
[19,]  14     14  1.3333333
[20,]  15      1 -0.6666667
```

5.4 Generating an A-inverse matrix using ainverse()

ainverse() uses the method of Meuwissen and Luo (1992) to compute the inverse relationship matrix directly from the pedigree. A complete description of the arguments and return value of a call to ainverse() is given in the ASReml-R Package Reference available at http://asreml.org under

Resources > ASReml docs and on the VSN International website https://www.vsni.co.uk.

5.5 Using Pedigree and G-inverse objects

Putting it all together, an analysis of *average daily gain* (y3 in harvey.dat) using the pedigree harvey.ped can be obtained from:

```
harvey.ped <- read.table("harvey.ped", header = T, as.is = T)

harvey <- asreml.read.table("harvey.dat", header = T, as.is = T)

harvey.ai <- ainverse(harvey.ped)

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

```
Model fitted using the gamma parameterization.
ASReml 4.1.0 Mon Aug 12 10:33:53 2019
         LogLik        Sigma2     DF     wall    cpu
 1      -239.696       663.731     62 10:33:53    0.0
 2      -239.347       591.895     62 10:33:53    0.0
 3      -238.932       433.351     62 10:33:53    0.0
 4      -238.842       331.737     62 10:33:53    0.0
 5      -238.831       284.401     62 10:33:53    0.0
 6      -238.831       273.598     62 10:33:53    0.0
```

The variance components are given in:

```
summary(adg0.asr)$varcomp
```

```
                    component std.error  z.ratio bound %ch
vm(Calf, harvey.ai)  500.3245  498.1786 1.004307     P 0.3
units!R              273.5984  409.7263 0.667759     P 0.0
```

and the first 15 E-BLUP estimates are:

```
head(adg0.asr$coef$random, 15)
```

```
                                    bu
vm(Calf, harvey.ai)_Sire_1  11.0207260
vm(Calf, harvey.ai)_Sire_2 -17.6304408
vm(Calf, harvey.ai)_Sire_3   6.6097149
vm(Calf, harvey.ai)_Sire_4  -8.0148417
vm(Calf, harvey.ai)_Sire_5   8.0148417
vm(Calf, harvey.ai)_Sire_6   8.4779824
vm(Calf, harvey.ai)_Sire_7   0.4442278
vm(Calf, harvey.ai)_Sire_8 -26.9512810
vm(Calf, harvey.ai)_Sire_9  18.0290709
vm(Calf, harvey.ai)_101     -7.2985297
vm(Calf, harvey.ai)_102     16.3863802
vm(Calf, harvey.ai)_103      2.5220427
vm(Calf, harvey.ai)_104     -6.7208489
```

```
vm(Calf, harvey.ai)_105     12.3426151
vm(Calf, harvey.ai)_106     17.5417417
```

This is an example of an individual animal model (I) estimating additive variance, σ^2_A, and an animal model residual, σ^2_{eI}. In this example we see the estimate of σ^2_A is 500.3245 and the estimate of σ^2_{eI} is 273.5984. The resulting variance matrix has terms $\sigma^2_A + \sigma^2_{eI}$, the phenotypic variance, in the diagonal elements. The only non-diagonal terms (1/4) in the relationship matrix occur between calves with the same sire (half-sibs) and so their covariance is $(1/4)\sigma^2_A$. Because of this relatively simple covariance structure an equivalent sire model can be fitted estimating a sire variance, σ^2_S and a sire model residual, σ^2_{eS}. In this model the phenotypic variance is given by $\sigma^2_S + \sigma^2_{eS}$ and the covariance between half-sibs is σ^2_S. We therefore expect $\sigma^2_A + \sigma^2_{eI} = \sigma^2_S + \sigma^2_{eA}$ and $\sigma^2_A = 4\sigma^2_S$ so that $\sigma^2_{eI} = \sigma^2_{eS} - 3\sigma^2_S$. An alternative and instructive way of deriving the sire model is to consider a reduced animal model (RA). The RA model was first introduced by Quass and Pollack (1980) to obtain predicted additive genetic effects without the need to set up equations for all individuals. In this case the additive genetic effects for animals without offspring u_{Acalf} are written in terms of their parental effects, u_{Adam}, and a mendelian sampling $u_{Amencalf}$ term. In our case the calves have no offspring so

$$u_{Acalf} = 0.5u_{Asire} + 0.5u_{Adam} + u_{Amencalf}.$$

The terms $u_{Amencalf}$ have additive variance $0.5\sigma^2_A$ so that u_{Acalf} has variance σ^2_A. In this case a further simplification occurs because the dams are unknown and we combine the last two terms to give $u_{Acalf} = 0.5u_{Asire} + u_{Arescalf}$ with $u_{Arescalf}$ having a variance $0.75\sigma^2_A$. The sire model takes this a stage further and has a sire model sire effect $u_{Rsire} = 0.5u_{Asire}$ and hence $\sigma^2_S = 0.25\sigma^2_A$. Further, the two residuals are combined to give

$$e_{Srescalf} = a_{Arescalf} + e_{Arescalf}$$

and $\sigma^2_{eS} = \sigma^2_{eI} - 0.75\sigma^2_A$. This model is now fitted:

```
asreml.options(gammaPar = TRUE)
adg1.asr <- asreml(y3 ~ Line, random = ~Sire, data = harvey)
```

```
Model fitted using the gamma parameterization.
ASReml 4.1.0 Mon Aug 12 10:33:54 2019
          LogLik        Sigma2     DF     wall    cpu
 1      -238.993       672.057     62 10:33:54    0.0
 2      -238.915       664.698     62 10:33:54    0.0
 3      -238.833       650.350     62 10:33:54    0.0
 4      -238.831       647.863     62 10:33:54    0.0
```

gives

```
summary(adg1.asr)$varcomp
```

```
        component std.error   z.ratio bound %ch
Sire     124.8934  124.9705 0.9993829     P 0.2
units!R  647.8634  122.3426 5.2954833     P 0.0
```

and the 9 Sire E-BLUP estimates are:

```
head(adg1.asr$coef$random, 9)
```

```
                    bu
Sire_Sire_1   5.5094593
Sire_Sire_2  -8.8136173
Sire_Sire_3   3.3041580
Sire_Sire_4  -4.0066901
Sire_Sire_5   4.0066901
Sire_Sire_6   4.2381029
Sire_Sire_7   0.2220591
Sire_Sire_8 -13.4731042
Sire_Sire_9   9.0129422
```

We see the final `logl` is the same in the two models (-238.831) and the E-BLUPs for sires in the individual animal model are twice those in the sire model. We see the estimate of σ^2_S is 124.8934 and the estimate of σ^2_{eS} is 647.8634. These convert to give estimates of $\sigma^2_A = 4\sigma^2_S = 4(124.8934) = 499.5736$ and $\sigma^2_{eI} = \sigma^2_{eS} - 3\sigma^2_S = 647.8634 - 3(124.8934) = 273.1562$ which agree well with the estimates derived from the animal model of $\sigma^2_A = 500.3245$ and $\sigma^2_{eI} = 273.5984$. We note that if the sire variance is over estimated, perhaps by not fitting all the appropriate fixed effects, the estimate of the animal residual variance found from conversion of the sire model estimates might be negative. If an animal model is fitted ASReml-R does not allow negative residual variances and the estimate will tend to zero. Estimates of the additive variance can be found by constraining the residual variance to a small positive variance. User specified general inverse matrices are included in an analysis in the same way. Consider an easily verified example where we define a general inverse matrix for Sire in the Harvey data as $(0.5)\boldsymbol{I}_9$ and estimate a scaled sire variance σ^2_{S2}. We expect $\sigma^2_S \boldsymbol{I}_9 = \sigma^2_{S2}(0.5)^{-1}\boldsymbol{I}_9 = 2\sigma^2_{S2}\boldsymbol{I}_9$ so $\sigma^2_{S2} = \sigma^2_S/2$.

```
sire.giv <- data.frame(row = seq(1, 9), column = seq(1, 9), value = rep(0.5, 9))
attr(sire.giv, "INVERSE") <- TRUE
attr(sire.giv, "rowNames") <- paste("Sire_", seq(1, 9), sep = "")
asreml.options(gammaPar = TRUE)
adg2.asr <- asreml(y3 ~ Line, random = ~vm(Sire, sire.giv), data = harvey)
```

```
Model fitted using the gamma parameterization.
ASReml 4.1.0 Mon Aug 12 10:33:56 2019
          LogLik        Sigma2     DF     wall    cpu
 1      -238.831       646.480     62 10:33:56    0.0
 2      -238.831       646.907     62 10:33:56    0.0
 3      -238.831       647.832     62 10:33:56    0.0
```

gives

```
summary(adg2.asr)$varcomp
```

```
                  component std.error   z.ratio bound %ch
vm(Sire, sire.giv)  62.44299  62.52914 0.9986223     P 0.1
units!R            647.83182 122.33371 5.2956116     P 0.0
```

In this case $\sigma^2_{S2} = 62.44299$ which is in good agreement with the converted value $(0.5)\sigma^2_S = 124.8934/2 = 62.4467$ from the previous analysis.

Alternatively, we can fit the equivalent model by including a matrix $2\boldsymbol{I}_9$ leading to the same log-likelihood and estimates of variances as the previous analysis:

5.5 Using Pedigree and G-inverse objects

```
sire.mat <- data.frame(row = seq(1, 9), column = seq(1, 9), value = rep(2, 9))
sire.mat <- as.matrix(sire.mat)
attr(sire.mat, "rowNames") <- paste("Sire_", seq(1, 9), sep = "")
adg3.asr <- asreml(y3 ~ Line, random = ~vm(Sire, sire.mat), data = harvey)
```

```
Model fitted using the gamma parameterization.
ASReml 4.1.0 Mon Aug 12 10:33:58 2019
          LogLik        Sigma2     DF     wall    cpu
 1      -238.831       646.480     62 10:33:58    0.0
 2      -238.831       646.907     62 10:33:58    0.0
 3      -238.831       647.832     62 10:33:58    0.0
```

```
summary(adg3.asr)$varcomp
```

```
                   component std.error   z.ratio bound %ch
vm(Sire, sire.mat)  62.44299  62.52914 0.9986223     P 0.1
units!R            647.83182 122.33371 5.2956116     P 0.0
```

Most variance matrices are positive definite so every linear combination of effects has a positive variance. However, there are situations in which a variance might be zero, for example, with a dominance model and a fully inbred individual. To illustrate this, we assume we have an extra individual unrelated to the first nine with zero variance so that the variance matrix has 10 rows and columns with nine diagonal elements being 2 and the last being zero. This matrix is positive semi-definite and to allow ASReml-R to continue using Lagrangian multiplies we indicate this by adding as a singularity code PSD. If the relationship matrix is read in as a vector

```
ped.vec <- c(2, 0, 2, 0, 0, 2, 0, 0, 0, 2, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0,
    0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0)
attr(ped.vec, "rowNames") <- paste("Sire_", seq(1, 10), sep = "")
adg4.asr <- asreml(y3 ~ Line, random = ~vm(Sire, ped.vec, "PSD"), data = harvey)
```

```
Note:  ped.vec assumed a relationship (not inverse) structure.
Note:  ped.vec assumed a relationship (not inverse) structure.
```

```
Warning in asreml(y3 ~ Line, random = ~vm(Sire, ped.vec, "PSD"), data = harvey):  ped.vec:
Zero pivot in row 10.
```

```
Warning in asreml(y3 ~ Line, random = ~vm(Sire, ped.vec, "PSD"), data = harvey):  ped.vec:
Zero pivot in row 11.
```

```
Model fitted using the gamma parameterization.
ASReml 4.1.0 Mon Aug 12 10:33:59 2019
          LogLik        Sigma2     DF     wall    cpu
 1      -238.831       646.480     62 10:33:59    0.0
 2      -238.831       646.907     62 10:33:59    0.0
 3      -238.831       647.832     62 10:33:59    0.0
```

gives

```
summary(adg4.asr)$varcomp
```

```
                        component std.error   z.ratio bound %ch
vm(Sire, ped.vec, "PSD")  62.44299  62.52914 0.9986223     P 0.1
units!R                  647.83182 122.33371 5.2956116     P 0.0
```

Again, if we have clones then the variance matrix can be positive semi definite, so if individual 10 was a clone of 9, reading in the relationship matrix as a sparse matrix:

```
spped.mat <- data.frame(Row = c(1, 2, 3, 4, 5, 6, 7, 8, 9, 10), Column = c(1, 2, 3, 4, 5,
    6, 7, 8, 9, 9), Value = c(1, 1, 1, 1, 1, 1, 1, 1, 1, 1))
spped.mat <- as.matrix(spped.mat)
attr(spped.mat, "rowNames") <- paste("Sire_", seq(1, 10), sep = "")
adg5.asr <- asreml(y3 ~ Line, random = ~vm(Sire, spped.mat, "NSD"), data = harvey)

Warning in checkPSD(model.terms$mixed$Vars):  smisg:  1 negative pivots in spped.mat.

Warning in asreml(y3 ~ Line, random = ~vm(Sire, spped.mat, "NSD"), data = harvey):
spped.mat:  Zero pivot in row 10; non-zero element in column 9

Warning in asreml(y3 ~ Line, random = ~vm(Sire, spped.mat, "NSD"), data = harvey):
spped.mat:  Zero pivot in row 9.

Warning in asreml(y3 ~ Line, random = ~vm(Sire, spped.mat, "NSD"), data = harvey):
spped.mat:  Negative pivot in row 10.

Model fitted using the gamma parameterization.
ASReml 4.1.0 Mon Aug 12 10:34:01 2019
         LogLik        Sigma2     DF     wall    cpu
 1      -238.993       672.057     62 10:34:01    0.0
 2      -238.915       664.698     62 10:34:01    0.0
 3      -238.833       650.350     62 10:34:01    0.0
 4      -238.831       647.863     62 10:34:01    0.0
```

gives

```
summary(adg5.asr)$varcomp
```

```
                          component std.error   z.ratio bound %ch
vm(Sire, spped.mat, "NSD")  124.8934  124.9705 0.9993829     P 0.2
units!R                     647.8634  122.3426 5.2954833     P 0.0
```

If the relationship matrix is estimated then it might be non-singular indefinite (positive and negative roots). ND indicates to ASReml-R to ignore the indefinite condition and proceed. If the matrix is singular indefinite (positive, zero and negative roots) NSD indicates to ASReml-R to proceed ignoring the indefinite condition and using Lagrangian multipliers to process the matrix.

5.6 Linking a relationship matrix to regressor variables

See Section 4.3.6.1 for details on linking a relationship matrix to regressor variables.

6 Prediction from the linear model

6.1 Introduction

Prediction is the process of forming a linear function of the vector of fixed and random effects in the linear model to obtain an estimated or 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.

Some predict methods require as input a data frame of the factor levels and variate values used to fit the model, augmented by new points for which predictions are required. This approach has limitations; for example, it does not lend itself easily to the notion of averaging over particular factors in the model to form predictions.

The approach to prediction described here is a generalization of that of Lane and Nelder (1982) who consider fixed effects models only. They form fitted values for all combinations of the explanatory variables in the model, then take marginal means across the explanatory variables not relevant to the current prediction. Our case is more general in that random effects can be fitted in mixed models. A full description can be found in Gilmour et al. (2004) and Welham et al. (2004).

Random factor terms may contribute to predictions in several ways. They may be evaluated at a given value(s) specified by the user, they may be averaged over, or they may be omitted from the fitted values used to form the prediction. Averaging over the set of random effects gives a prediction specific to the random effects observed. We describe this as a *conditional* prediction. Omitting the term from the model produces a prediction at the population average (zero), that is, substituting the assumed population mean for an unknown random effect. We call this a *marginal* prediction. Note that in any prediction, some terms may be evaluated as conditional and others at marginal values, depending on the aim of prediction.

For fixed factors there is no pre-defined population average, so there is no natural interpretation for a prediction derived by omitting a fixed term from the fitted values. Averages must therefore be taken over all the levels present to give a sample specific average, or value(s) must be specified by the user.

For covariate terms (fixed or random) the associated effect represents the coefficient of a linear

trend in the data with respect to the covariate values. These terms should be evaluated at a given value of the covariate, or averaged over several given values. Omission of a covariate from the predictive model is equivalent to predicting at a zero covariate value, which is often inappropriate.

Interaction terms constructed from factors generate an effect for each combination of the factor levels, and behave like single factor terms in prediction. Interactions constructed from covariates fit a linear trend for the product of the covariate values and behave like a single covariate term. An interaction of a factor and a covariate fits a linear trend for the covariate for each level of the factor. For both fixed and random terms, a value for the covariate must be given, but the factor levels may be evaluated at a given level, averaged over or (for random terms only) omitted.

Before considering some examples in detail, it is useful to consider the conceptual steps involved in the prediction process. Given the explanatory variables used to define the linear (mixed) model, the four main steps are:

1. Choose the explanatory variable(s) and their respective value(s) for which predictions are required; the variables involved will be referred to as the ***classify*** set and together define the multiway table to be predicted.

2. Determine which variables should be averaged over to form predictions. The values to be averaged over must also be defined for each variable; the variables involved will be referred to as the ***averaging*** set. The combination of the classify set with these averaging variables defines a multiway hyper-table. Note that variables evaluated at only one value, for example, a covariate at its mean value, can be formally introduced as part of the classifying or averaging set.

3. Determine which terms from the linear mixed model are to be used in forming predictions for each cell in the multiway hyper-table in order to give appropriate conditional or marginal predictions.

4. Choose the weights to be used when averaging cells in the hyper-table to produce the multiway table to be reported.

Note that after steps 1 and 2 there may be some explanatory variables in the fitted model that do not classify the hyper-table. These variables occur in terms that are ignored when forming the predicted values. It was concluded above that fixed terms could not sensibly be ignored in forming predictions, so that variables should only be omitted from the hyper-table when they only appear in random terms. Whether terms derived from these variables should be used when forming predictions depends on the application and aim of the prediction.

The main difference in this prediction process compared to that described by Lane and Nelder (1982) is the choice of whether to include or exclude model terms when forming predictions. In linear models, since all terms are fixed, terms not in the classify set must be in the averaging set.

6.2 The predict method

The predict method is detailed in the ASReml-R Package Reference obtained in R by typing `help(asreml)`. This document is available at `http://asreml.org` under `Resources > ASReml docs` and on the VSN International website `https://www.vsni.co.uk`. A simple example is the prediction of variety means from fitting model 2a (Section 4.1) to the NIN field trial data. Recall that a randomised

block model with random replicate effects is fitted to this data by:

```
> asreml.options(gammaPar = TRUE)
> rcb.asr <- asreml(yield ~ Variety, random = ~idv(Replicate), residual = ~idv(units),
    na.action = na.method(x = "include"), data = nin89)
```

```
Model fitted using the gamma parameterization.
ASReml 4.1.0 Mon Aug 12 10:34:04 2019
          LogLik        Sigma2     DF     wall    cpu
 1      -454.807       50.3285    168 10:34:04    0.0
 2      -454.663       50.1198    168 10:34:04    0.0
 3      -454.532       49.8684    168 10:34:04    0.0
 4      -454.472       49.6374    168 10:34:04    0.0
 5      -454.469       49.5854    168 10:34:04    0.0
```

A table of means classified by `Variety` can be obtained from:

```
> rcb.pv <- predict(rcb.asr, classify = "Variety")
```

```
Model fitted using the gamma parameterization.
ASReml 4.1.0 Mon Aug 12 10:34:05 2019
          LogLik        Sigma2     DF     wall    cpu
 1      -454.469       49.5824    168 10:34:05    0.1
 2      -454.469       49.5824    168 10:34:05    0.0
 3      -454.469       49.5824    168 10:34:05    0.0
```

```
> names(rcb.pv)
```

```
[1] "pvals" "avsed"
```

A component named `pvals` is included in the asreml object:

```
> head(rcb.pv$pvals)
```

```
Notes:
- The predictions are obtained by averaging across the hypertable
  calculated from model terms constructed solely from factors in
  the averaging and classify sets.
- Use 'average' to move ignored factors into the averaging set.
- The ignored set: Replicate
```

```
    Variety predicted.value std.error    status
1  ARAPAHOE         29.4375  3.855687 Estimable
2     BRULE         26.0750  3.855687 Estimable
3  BUCKSKIN         25.5625  3.855687 Estimable
4   CENTURA         21.6500  3.855687 Estimable
5 CENTURK78         30.3000  3.855687 Estimable
6  CHEYENNE         28.0625  3.855687 Estimable
```

as well as the average standard error of difference between the predicted means:

```
> rcb.pv$avsed
```

```
 overall
4.979075
```

6.2.1 The prediction process

Predictions are formed as an extra process in the final iteration. predict.asreml() parses the argument list and calls update.asreml() using the final parameter estimates in the required asreml object. Additional options for predict.asreml() can be set in asreml.options(), such as requesting extra memory, adding spline predict points or controlling the number of additional iterations, bound by the rules of update.asreml().

By default, 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. Covariates grouped into a single term using the grp() model function) are treated as covariates.

Special model terms `mv` and `units` are always ignored.

Prediction at particular values of a covariate or particular levels of a factor is achieved by:

1. including the variables in the *classify* set and specifying any non-default values at which predictions are to be made by using the `levels` argument.

2. specifying the averaging set. The default averaging set is those explanatory variables involved in fixed effect model terms that are not in the classifying set. By default, variables that only define random model terms are ignored. The `average` argument allows these variables to be added to the default averaging set.

3. determining the linear model terms to use in prediction. The default rule is that all model terms based entirely on the classifying and averaging set are used. The `use` and `ignore` arguments allow this default set of model terms to be modified by adding or removing terms, respectively. The `onlyuse` argument explicitly specifies the model terms to use, ignoring all others. The argument `except` explicitly specifies the model terms not to use, including all others. These arguments may implicitly modify the averaging set by including variables defining terms in the predicted model not in the classify set. It is sometimes easier to specify the classify set and the prediction linear model and allow ASReml-R to construct the averaging set.

4. choosing the weights for forming means over dimensions in the hyper-table. The default is to average over the specified levels but the `average` argument can be used to specify weights to be used in averaging over a factor.

For example,

```
> obj.asr <- asreml(yield ~ Site + Variety, random = ~Site:Variety + at(Site):Block, ...)
> pbj.pv <- predict(obj.asr, classify = "Variety")
```

puts `Variety` in the classify set, `Site` in the averaging set and `Block` in the ignore set. Consequently, the `Site`×`Variety` hyper-table is formed from from model terms `Site`, `Variety` and `Site:Variety` but ignoring all terms in `at(Site):Block`, and then averaging across sites to produce variety predictions.

6.3 Aliasing

There are often situations in which the fixed effects design matrix $\boldsymbol{X}$ is not of full column rank. These can be classified according to the cause of aliasing:

1. linear dependencies among the model terms due to over-parameterisation of the model.

2. no data present for some factor combinations so that the corresponding effects cannot be estimated.

3. linear dependencies due to other, usually unexpected, structure in the data.

The first type of aliasing is imposed by the parameterisation chosen and can be determined from the model. The second type of aliasing can be detected when setting up the design matrix for parameter estimation (which may require revision of imposed constraints). The third type can then be detected during the absorption of the mixed model equations. Dependencies (aliasing) can be dealt with in several ways and ASReml-R checks that predictions are of estimable functions in the sense defined by Searle (1971, p160) and are invariant to the constraint method used.

Normally ASReml-R does not return predictions of non-estimable functions but the `aliased` argument can be used to control this for each predict table. However, using `aliased` is rarely a satisfactory solution. Failure to report predicted values normally means that the prediction is averaging over some cells of the hyper-table that have no information and therefore cannot be averaged in a meaningful way. Appropriate use of the `average` or `present` arguments will usually resolve the problem. The `present` argument enables the construction of means by averaging only the estimable cells of the hyper-table. It is regularly used for nested factors, for example *locations* nested in *regions*.

6.4 Complicated weighting

Generally, when forming a prediction table, it is necessary to average over (or ignore) some potential dimensions of the prediction table. By default, ASReml-R uses equal weights ($1/f$ for a factor with f levels). More complicated weighting is achieved by using the `average` argument to set specific (unequal) weights for each level of a factor. However, sometimes the weights to be used need to be defined with respect to two or more factors. The simplest case is when there are missing cells and weighting is equal for those cells in a multiway table that are present; this is achieved by using the `present` argument. This is further generalized by allowing weights for use by the `present` averaging process via a named component `prwts` of the `present` list.

The factors in the table of weights are specified with the `present` argument and the table of weights with the `prwts` component of the `present` list. There may be a maximum of two independent lists of factors in the `present` list, and, if specified `prwts` applies to the first list only. The order of factors in the tables of weights must correspond to the order in the `present` list with later factors

nested within preceding factors. Check the output to ensure that the values in the tables of weights are applied in the correct order.

Consider a rather complicated example from a rotation experiment conducted over several years. This particular analysis followed the analysis of the daily live weight gain per hectare of the sheep grazing the plots. There were periods when no sheep grazed. Different flocks grazed in the different years. Daily liveweight gain was assessed between 5 and 8 times in the various years. To obtain a measure of total productivity in terms of sheep liveweight, we need to weight the daily per sheep figures by the number of sheep grazing days per month. Treatment effects for each year can be obtained from:

```
> predict(obj.asr, classify = "year:crop:pasture:lime",
    levels = list(year=1,crop=1),
    average = list(month=c(56,55,56,53,57,63 rep(0,6))))

> predict(obj.asr, classify = "year:crop:pasture:lime", levels = list(year = 2, crop = 1),
    average = list(month = c(36, 0, 0, 53, 23, 24, 54, 54, 43, 35, 0, 0)))

> predict(obj.asr, classify = "year:crop:pasture:lime", levels = list(year = 3, crop = 1),
    average = list(month = c(70, 0, 21, 17, 0, 0, 0, 70, 0, 0, 53, 0)))

> predict(obj.asr, classify = "year:crop:pasture:lime", levels = list(year = 4, crop = 1),
    average = list(month = c(53, 56, 22, 92, 19, 44, 0, 0, 36, 0, 0, 49)))

> predict(obj.asr, classify = "year:crop:pasture:lime", levels = list(year = 5, crop = 1),
    average = list(month = c(0, 22, 0, 53, 70, 22, 0, 51, 16, 51, 0, 0)))
```

and averages over years from:

```
> predict(obj.asr, classify="crop:pasture:lime",
    levels = list(crop=1),
    present = list(c("year","month"),
    prwts=c(56,55,56,53,57,63,0,0,0,0,0,0,
    36,0,0,53,23,24,54,54,43,35,0,0,
    70,0,,21,17,0,0,0,70,0,0,53,0,
    53,56,22,92,19,44,0,0,36,0,0,49,
    0,22,0,53,70,22,0,51,16,51,0,0}/5))
```

Both sets of predict() calls are given to show how the weights were derived and used. Notice that the order in `c("year", "month")` implies that the weight coefficients are presented in standard order with the levels for months cycling within levels for years.

6.5 Further examples

Predict variety means from an RCB analysis of the NIN field trial data.

```
> nin89.asr <- asreml(fixed = yield ~ Variety, random = ~Rep, data = nin89)
> nin89.pv <- predict(nin89.fm, classify = "Variety")
```

Variety means from the NIN field trial data in the presence of a covariate `x`.

6.5 Further examples

```
> nin89.asr <- asreml(fixed = yield ~ Variety + x, random = ~Rep, data = nin89)
> nin89.pv <- predict(nin89.fm, classify = "Variety")
```

will predict variety means at the average of `x` ignoring random replicate effects.

Variety means from the NIN field trial data at a specified value of `x`

```
> nin89.asr <- asreml(fixed = yield ~ Variety + x + Rep, data = nin89)
> nin89.pv <- predict(nin89.fm, classify = "Variety:x", levels = list(x = 2))
```

predicts variety means at `x`=2, averaged over fixed replicate effects.

Variety effects from an across site analysis

```
> obj.asr <- asreml(fixed = yield ~ Variety, random = ~Variety:Site, data = ...)
> obj.pv <- predict(sm, classify = "Variety")
```

predicts variety means ignoring the `site:variety` term while

```
> obj.pv <- predict(sm, classify="Variety", average=list(Site=NULL))}
```

forms the hyper-table based on `Site` and `Variety` with each cell formed from linear combinations of `Variety` and `Variety:Site` effects; `Variety` predictions are then formed from averages across `Site` levels.

Predict trait means for each team for the Orange wether trial

```
> orange.asr <- asreml(cbind(gfw, fdiam) ~ trait + trait:Year, random = ~trait:Team, data =
    orange)
> orange.pv <- predict(orange.asr, classify = "trait:Team")
```

forms the hyper-table for each trait based on `Year` and `Team` with each linear combination in each cell of the hyper-table for each trait using `Team` and `Year` effects. `Team` predictions are produced by averaging over years.

7 Examples

7.1 Introduction

This section considers the analysis of several examples to illustrate the capabilities of ASReml-R in the context of analysing real data-sets. We discuss some of the components returned from ASReml-R and indicate when potential problems may occur. Statistical concepts and issues are discussed as necessary but we stress that the analyses are only illustrative.

7.2 Split Plot Design

The first example is the analysis of a split plot design originally presented by Yates (1935). The experiment was conducted to assess the effects on yield of three oat varieties (Golden Rain, Marvellous and Victory) with four levels of nitrogen application (0, 0.2, 0.4 and 0.6 cwt/acre). The field layout consisted of six blocks (labelled I, II, III, IV, V and VI) with three whole-plots per block each split into four sub-plots. The three varieties were randomly allocated to the three whole-plots while the four levels of nitrogen application were randomly assigned to the four sub-plots within each whole-plot. The data is in Table 7.1.

Table 7.1: A split-plot field trial of oat varieties and nitrogen application

		Nitrogen			
Block	Variety	0.0cwt	0.2cwt	0.4cwt	0.6cwt
I	GR	111	130	157	174
	M	117	114	161	141
	V	105	140	118	156
II	GR	61	91	97	100
	M	70	108	126	149
	V	96	124	121	144
III	GR	68	64	112	86
	M	60	102	89	96
	V	89	129	132	124
IV	GR	74	89	81	122
	M	64	103	132	133
	V	70	89	104	117
V	GR	62	90	100	116
	M	80	82	94	126
	V	63	70	109	99
VI	GR	53	74	118	113
	M	89	82	86	104
	V	97	99	119	121

A standard analysis of these data recognises the two basic elements inherent in the experiment:

1. the stratification of the experiment units, that is the *blocks*, *whole-plots* and *sub-plots*, and

2. the treatment structure that is superimposed on the experimental material.

The latter is of prime interest in the presence of stratification. The aim of the analysis is to examine the importance of the treatment effects while accounting for the stratification and restricted randomisation of the treatments to the experimental units.

The function calls to initially create a data frame and perform the standard split-plot analysis in ASReml-R are given below. The variate/factor names are specified in the header line of `oats.txt`, with factor names beginning with a capital letter. The function `asreml.read.table()` recognises this convention and automatically creates the factors in the data frame:

```
oats <- asreml.read.table("../UKMay16/oats.txt", header = T)
asreml.options(gammaPar = TRUE)
oats.asr <- asreml(fixed = yield ~ Variety + Nitrogen + Variety:Nitrogen, random =
    ~idv(Blocks) + idv(Blocks):idv(Wplots), residual = ~idv(units), data = oats)
```

The fields in the `oats` data frame are:

```
> names(oats)

[1] "Blocks"   "Nitrogen" "Subplots" "Variety"  "Wplots"   "yield"
```

The first five are factors describing the stratification, or experiment design, and applied treatments. The standard split plot analysis is achieved by fitting terms `Block` and `Blocks:Wplots` as random effects. It is not necessary to specify `Blocks:Wplot:Subplots` as these three factors uniquely define the experimental units. The fixed effects include the main effects of both `Variety` and `Nitrogen` and their interaction.

The variance components are:

```
> summary(oats.asr)$varcomp

                     component std.error  z.ratio bound %ch
Blocks!Blocks         214.4906 168.66037 1.271731     P 0.1
Blocks:Wplots!Blocks  106.0686  67.88201 1.562543     P 0.0
units!R               177.0946  37.33601 4.743266     P 0.0
```

For simple variance component models such as the above, the default parameterisation for the variance parameters is as the ratio to the residual variance. Thus ASReml-R returns the REML estimate of the variance parameters on the `gamma` and `sigma` scale for each term in the random model.

The default synopsis for testing fixed effects in ASReml-R is a table of incremental Wald tests (see Section 3.14):

```
> wald(oats.asr)

Wald tests for fixed effects.
Response: yield
```

```
Terms added sequentially; adjusted for those above.

                 Df Sum of Sq Wald statistic Pr(Chisq)
(Intercept)       1     43444        245.312    <2e-16 ***
Variety           2       526          2.970    0.2264
Nitrogen          3     20021        113.050    <2e-16 ***
Variety:Nitrogen  6       322          1.817    0.9358
residual (MS)             177
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

In this example there are four terms included in the summary. The overall mean `(Intercept)` is included though it is of no interest for these data. The tests are sequential, that is the effect of each term is assessed by the change in sums of squares achieved by adding the term to the current model, given those terms appearing above the current term are already included. For example, the effect of `Nitrogen` is assessed by calculating the change in sums of squares for the two models `(Intercept)+Variety+Nitrogen` and `(Intercept)+Variety`. No refitting occurs, that is the variance parameters are held constant at the REML estimates obtained from the currently specified fixed model.

The usual ANOVA divides into three strata, with treatment effects separating into different strata as a consequence of the balanced design and the confounding of main effects of `Variety` with whole-plots. It is straightforward to derive the ANOVA estimates of the stratum variances from the above REML estimates. That is,

$$\begin{aligned} blocks &= 12\hat{\sigma}_b^2 + 4\hat{\sigma}_w^2 + \hat{\sigma}^2 = 3175.1 \\ blocks.wplots &= 4\hat{\sigma}_w^2 + \hat{\sigma}^2 = 601.3 \\ residual &= \hat{\sigma}^2 = 177.1 \end{aligned}$$

The incremental Wald tests have an asymptotic χ^2 distribution, with degrees of freedom (df) given by the number of estimable effects (the number in the `df` column). The denominator degrees of freedom for testing fixed effects and approximate stratum variances are returned by:

```
> oats.wld <- wald(oats.asr, denDF = "default")

Model fitted using the gamma parameterization.
ASReml 4.1.0 Mon Aug 12 10:34:09 2019
          LogLik        Sigma2     DF     wall    cpu
 1      -209.378       177.083     60 10:34:09    0.0
 2      -209.378       177.083     60 10:34:09    0.0
 3      -209.378       177.083     60 10:34:09    0.0

> oats.wld$Wald


Wald tests for fixed effects.
Response: yield

                 Df denDF   F.inc      Pr
(Intercept)       1     5 245.100 0.00002
```

```
Variety           2    10   1.485 0.27239
Nitrogen          3    45  37.690 0.00000
Variety:Nitrogen  6    45   0.303 0.93220
```

```
> oats.wld$stratumVariances
```

```
                     df  Variance Blocks!Blocks Blocks:Wplots!Blocks units!R
Blocks!Blocks         5 3175.0556            12                     4       1
Blocks:Wplots!Blocks 10  601.3306             0                     4       1
units!R              45  177.0833             0                     0       1
```

Determining the denominator degrees of freedom is straightforward for balanced designs, such as the split-plot design, but it is not so straightforward in unbalanced designs, such as the rat data set described in the next section.

Predicted means for the `Variety`, `Nitrogen` and `Variety:Nitrogen` effects can be obtained from the predict method in separate statements:

```
> oatsV.pv <- predict(oats.asr, classify = "Variety", sed = T)
```

```
> oatsN.pv <- predict(oats.asr, classify = "Nitrogen", sed = T)
```

```
> oatsVN.pv <- predict(oats.asr, classify = "Variety:Nitrogen", sed = T)
```

The latter returns an object `oatsVN.pv` with components `pvals`, `sed` and `avsed`: `pvals` contains the predicted means for the term defined in the `classify` argument and `sed` contains the full matrix of SEDs for this set of predictions (`Variety:Nitrogen` here):

```
> oatsVN.pv$pvals
```

```
Notes:
- The predictions are obtained by averaging across the hypertable
  calculated from model terms constructed solely from factors in
  the averaging and classify sets.
- Use 'average' to move ignored factors into the averaging set.
- The ignored set: Blocks,Wplots
```

```
       Variety Nitrogen predicted.value std.error    status
1  Golden_rain  0.2_cwt        98.50000  9.106977 Estimable
2  Golden_rain  0.4_cwt       114.66667  9.106977 Estimable
3  Golden_rain  0.6_cwt       124.83333  9.106977 Estimable
4  Golden_rain    0_cwt        80.00000  9.106977 Estimable
5   Marvellous  0.2_cwt       108.50000  9.106977 Estimable
6   Marvellous  0.4_cwt       117.16667  9.106977 Estimable
7   Marvellous  0.6_cwt       126.83333  9.106977 Estimable
8   Marvellous    0_cwt        86.66667  9.106977 Estimable
9      Victory  0.2_cwt        89.66667  9.106977 Estimable
10     Victory  0.4_cwt       110.83333  9.106977 Estimable
```

```
11     Victory  0.6_cwt       118.50000  9.106977 Estimable
12     Victory    0_cwt        71.50000  9.106977 Estimable
```

```
> oatsVN.pv$sed
```

```
12 x 12 Matrix of class "dspMatrix"
          [,1]     [,2]     [,3]     [,4]     [,5]     [,6]     [,7]     [,8]     [,9]
 [1,]       NA 7.682954 7.682954 7.682954 9.715025 9.715025 9.715025 9.715025 9.715025
 [2,] 7.682954       NA 7.682954 7.682954 9.715025 9.715025 9.715025 9.715025 9.715025
 [3,] 7.682954 7.682954       NA 7.682954 9.715025 9.715025 9.715025 9.715025 9.715025
 [4,] 7.682954 7.682954 7.682954       NA 9.715025 9.715025 9.715025 9.715025 9.715025
 [5,] 9.715025 9.715025 9.715025 9.715025       NA 7.682954 7.682954 7.682954 9.715025
 [6,] 9.715025 9.715025 9.715025 9.715025 7.682954       NA 7.682954 7.682954 9.715025
 [7,] 9.715025 9.715025 9.715025 9.715025 7.682954 7.682954       NA 7.682954 9.715025
 [8,] 9.715025 9.715025 9.715025 9.715025 7.682954 7.682954 7.682954       NA 9.715025
 [9,] 9.715025 9.715025 9.715025 9.715025 9.715025 9.715025 9.715025 9.715025       NA
[10,] 9.715025 9.715025 9.715025 9.715025 9.715025 9.715025 9.715025 9.715025 7.682954
[11,] 9.715025 9.715025 9.715025 9.715025 9.715025 9.715025 9.715025 9.715025 7.682954
[12,] 9.715025 9.715025 9.715025 9.715025 9.715025 9.715025 9.715025 9.715025 7.682954
         [,10]    [,11]    [,12]
 [1,] 9.715025 9.715025 9.715025
 [2,] 9.715025 9.715025 9.715025
 [3,] 9.715025 9.715025 9.715025
 [4,] 9.715025 9.715025 9.715025
 [5,] 9.715025 9.715025 9.715025
 [6,] 9.715025 9.715025 9.715025
 [7,] 9.715025 9.715025 9.715025
 [8,] 9.715025 9.715025 9.715025
 [9,] 7.682954 7.682954 7.682954
[10,]       NA 7.682954 7.682954
[11,] 7.682954       NA 7.682954
[12,] 7.682954 7.682954       NA
```

The `avsed` component gives the minimum, mean and maximum `SED`:

```
> oatsVN.pv$avsed
```

```
     min     mean      max
7.682954 9.160824 9.715025
```

The average SED is calculated from the average variance of differences and is given as:

```
> oatsVN.pv$avsed[2]
```

```
    mean
9.160824
```

7.3 Unbalanced nested design

This example illustrates some further aspects of testing fixed effects in linear mixed models. It differs from the previous split plot example in that it is unbalanced, so more care is required in assessing the significance of fixed effects.

The experiment was reported by Dempster et al. (1984) and was designed to compare the effect of three doses of an experimental compound (control, low and high) on the maternal performance of rats. Thirty female rats (`Dams`) were randomly split into three groups of 10 and each group randomly assigned to the three different doses. All pups in each litter were weighed. The litters differed both in total size and composition of males and females. Thus the additional covariate `littersize` was included in the analysis. The differential effect of the compound on male and female pups was also of interest.

Three litters had to be dropped from the experiment, which meant that one dose had only 7 dams. The analysis must account for the presence of between dam variation, but must also recognise the stratification of the experimental units (pups within litters) and the restricted randomisation of the doses to the dams. An indicative ANOVA decomposition for this experiment is given in Table 7.2.

Table 7.2: Rat data: ANOVA decomposition

stratum	decomposition	type	df or ne
(Intercept)		fixed	1
Dams			
	Dose	fixed	2
	littersize	fixed	1
	Dam	random	27
Dams:Pups			
	Sex	fixed	1
	Dose:Sex	fixed	2
error		random	

The `Dose` and `littersize` effects are implicitly tested against the residual dam variation, while the remaining effects are tested against the residual within litter variation. The asreml() call is:

```
> asreml.options(gammaPar = TRUE)
> rats.asr <- asreml(weight ~ littersize + Dose + Sex + Dose:Sex, random = ~idv(Dam),
    residual = ~idv(units), data = rats)
```

An abbreviated output from the asreml convergence trace removing iterations 2 to 5 is:

```
> rats.asr$trace[, (-2:-5)]

                    1           6           7           8
LogLik       74.2174175  87.2396114  87.2397915  87.2397915
Sigma2        0.1967003   0.1653687   0.1652991   0.1652991
DF          315.0000000 315.0000000 315.0000000 315.0000000
```

```
Dam!Dam       0.1000000   0.5827630   0.5867030   0.5866740
units!units   1.0000000   1.0000000   1.0000000   1.0000000
units!R       1.0000000   1.0000000   1.0000000   1.0000000
```

The tables of estimated variance parameters (from `summary()`) and Wald tests (from `wald()`) are:

```
> summary(rats.asr)$varcomp

             component  std.error  z.ratio bound %ch
Dam!Dam     0.09697668 0.03318630  2.92219     P   0
units!units 0.16529909         NA       NA     F   0
units!R     0.16529909 0.01367002 12.09209     P   0

> wald(rats.asr, denDF = "default", ssType = "conditional")$Wald

Model fitted using the gamma parameterization.
ASReml 4.1.0 Mon Aug 12 10:34:14 2019
          LogLik        Sigma2     DF     wall    cpu
 1       87.2398      0.165300    315 10:34:14    0.0
 2       87.2398      0.165300    315 10:34:14    0.0
 3       87.2398      0.165300    315 10:34:14    0.0


Wald tests for fixed effects.
Response: weight

            Df denDF  F.inc   F.con Margin      Pr
(Intercept)  1  32.0 9049.0 1099.00        0.00000
littersize   1  31.5   28.0   46.25      B 0.00000
Dose         2  23.9   12.2   11.51      A 0.00031
Sex          1 299.8   58.0   57.96      A 0.00000
Dose:Sex     2 302.1    0.4    0.40      B 0.67175
```

The incremental Wald tests indicate that the interaction between `Dose` and `Sex` is not significant. Since these tests are sequential then the test for the `Dose:Sex` term is appropriate as it respects marginality with both the main effects of dose and sex fitted before the inclusion of the interaction.

The conditional F-test helps assess the significance of the other terms in the model. It confirms `littersize` is significant after the other terms, that `Dose` is significant when adjusted for `littersize` and `Sex` but ignoring `Dose.Sex`, and that `Sex` is significant when adjusted for `littersize` and `Dose` but ignoring `Dose.Sex`. These tests respect marginality to the `Dose.Sex` interaction.

A plot of residuals vs fitted values is shown in Figure 7.1:

```
> plot(rats.asr, formula = resid(.) ~ fitted(.), fun = "xyplot")
```

Before proceeding we note the possibility of several outliers, in particular unit 66. The weight of this female rat, within litter 9 is only 3.68, compared to weights of 7.26 and 6.58 for two other female

sibling pups. This weight appears erroneous, but without knowledge of the actual experiment we retain the observation.

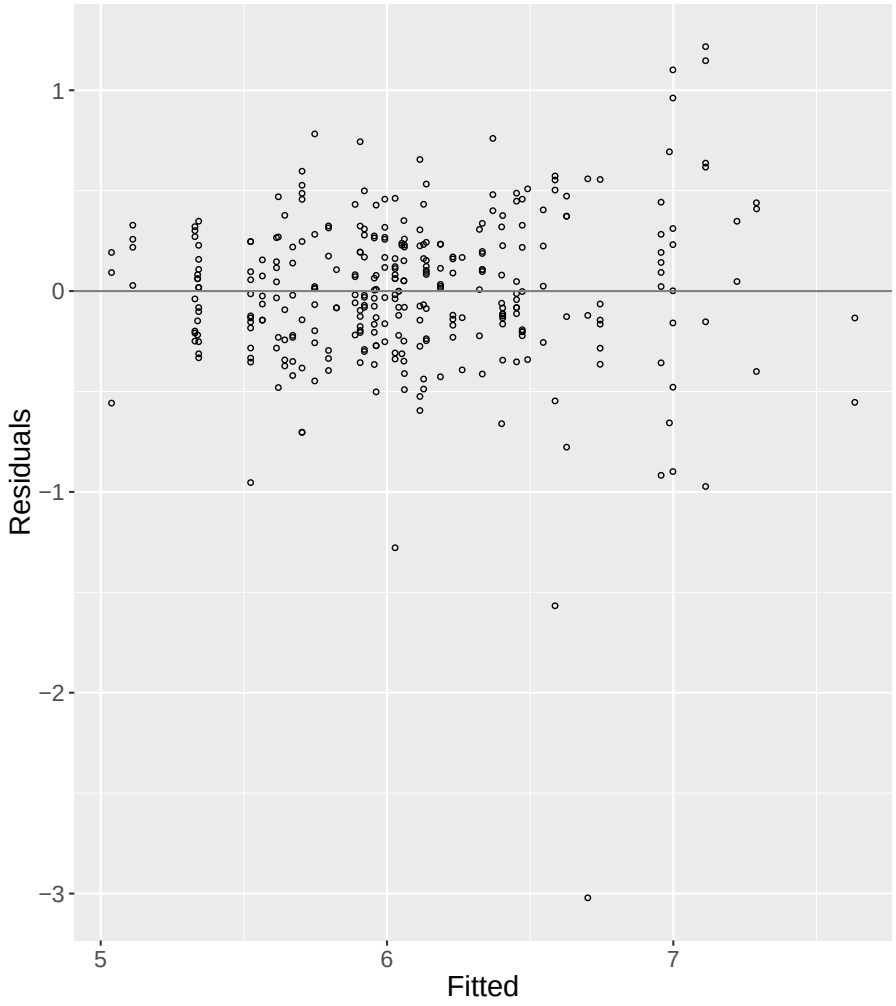

Figure 7.1: Residual plot for the rat data

We refit the model without the `Dose:Sex` term.

```
> asreml.options(gammaPar = TRUE)
> rats2.asr <- asreml(weight ~ littersize + Sex + Dose, random = ~idv(Dam), residual =
    ~idv(units), data = rats)
```

```
> summary(rats2.asr)$varcomp
```

```
             component  std.error   z.ratio bound %ch
Dam!Dam     0.09791765 0.03341525  2.930328     P   0
units!units 0.16452411         NA        NA     F   0
units!R     0.16452411 0.01356055 12.132553     P   0
```

```
> ww <- wald(rats2.asr, denDF = "default", ssType = "conditional")
```

```
> ww$Wald
```

```
Wald tests for fixed effects.
Response: weight

            Df denDF  F.inc   F.con Margin         Pr
(Intercept)  1  32.0 8981.0 1093.00        0.00000000
littersize   1  31.4   27.9   46.43      A 0.00000012
Sex          1 301.7   59.5   58.27      A 0.00000000
Dose         2  24.0   11.4   11.42      A 0.00032833
```

Note that the variance parameters are re-estimated, though there is little change from the previous analysis.

The impact of (wrongly) dropping `Dam` from this model is shown below:

```
> asreml.options(gammaPar = TRUE)
> rats3.asr <- asreml(weight ~ littersize + Dose + Sex, residual = ~idv(units), data =
    rats)
```

```
> ww <- wald(rats3.asr, denDF = "default", ssType = "conditional")
```

```
> ww$Wald
```

```
Wald tests for fixed effects.
Response: weight

            Df denDF F.inc  F.con Margin          Pr
(Intercept)  1   317 47080 3309.0        0.00000e+00
littersize   1   317    68  146.5      A 0.00000e+00
Dose         2   317    61   58.4      A 0.00000e+00
Sex          1   317    25   24.5      A 1.19915e-06
```

Even if a random term is not 'significant', it should not be dropped from the model if it represents a strata of the design as in this case. The impact of deleting `Dam` on the significance tests for the fixed effects is substantial and not surprising. This reinforces the importance of preserving the strata of the design when assessing the significance of fixed effects.

7.4 Sources of variability in unbalanced data

This example illustrates an approach to the analysis of unbalanced data where the main aim is to determine the sources of variation rather than assess the significance of imposed treatments. The data are taken from Cox and Snell (1981) and involve an experiment to examine the variability in the production of car voltage regulators. Standard production of regulators involves two steps: 1) Regulators are taken from the production line and passed to a setting station which adjusts the regulator to operate within a specified range of voltages, and, 2) from the setting station the regulator is then passed to a testing station where it is tested and returned if outside the required

range.

A total of 64 regulators was tested at four testing stations (`Teststat`). The voltage for individual regulators was set at a total of 10 setting stations (`Setstat`). A variable number of regulators (between 4 to 8) were set at each station. However, each regulator was tested at every testing station. The asreml() function call is:

```
voltage <- asreml.read.table("../examples/voltage.csv", header = T, sep = ",")
names(voltage)
```

```
> asreml.options(gammaPar = TRUE)
> voltage.asr <- asreml(voltage ~ 1, random = ~Setstat + Setstat:Regulatr + Teststat +
    Setstat:Teststat, residual = ~idv(units), data = voltage)
```

The factor `Regulatr` numbers the regulators within each setting station. Thus the term `Setstat:Regulatr` allows for differential effects of each regulator, while the other terms examine the effects of the setting and testing stations and possible interaction.

The estimated components of variance are:

```
> summary(voltage.asr)$varcomp
```

```
                    component    std.error   z.ratio bound %ch
Teststat         3.287141e-03 3.337459e-03 0.9849231     P   0
Setstat          1.193738e-02 8.811857e-03 1.3546953     P   0
Setstat:Teststat 5.175173e-09 5.323728e-10 9.7209570     B   0
Setstat:Regulatr 3.077791e-02 8.452247e-03 3.6413883     P   0
units!units      5.114166e-02           NA        NA     F   0
units!R          5.114166e-02 5.260970e-03 9.7209570     P   0
```

The convergence criterion was satisfied, however, the variance component estimate for the `Setstat:Teststat` term has been fixed at the boundary. The default constraint for variance components is to ensure that the REML estimate remains positive. If an update for any variance component results in a negative value then ASReml-R sets that variance component to a small positive value. If this occurs in subsequent iterations the parameter is fixed at the boundary. The default parameter constraints (**P**ositive for variance components) can be altered (to **U**nconstrained, for example) by changing the constraint code in the initial value list object(s) for random parameters, that is, the `R.param` and `G.param` arguments to asreml(). These lists are returned in the asreml object and are best accessed via the function `asreml.gammas.ed()`. In this example, the following sequence would achieve this:

```
> # Edit appropriate parameter code
> asreml.options(gammaPar = TRUE)
> voltage.asr <- asreml(voltage ~ 1, random = ~Setstat + Setstat:Regulatr + Teststat +
    Setstat:Teststat, residual = ~idv(units), G.param = temp$G.param, data = voltage)
```

though it would not generally be recommended for standard analyses.

```
> plot(voltage.asr)
```

includes a residual plot which indicates two unusual data values (Figure 7.2). These values are successive observations, 210 and 211, respectively, being testing stations 2 and 3 for setting station J, regulator 2. These observations will be retained for consistency with other analyses conducted by Cox and Snell (1981).

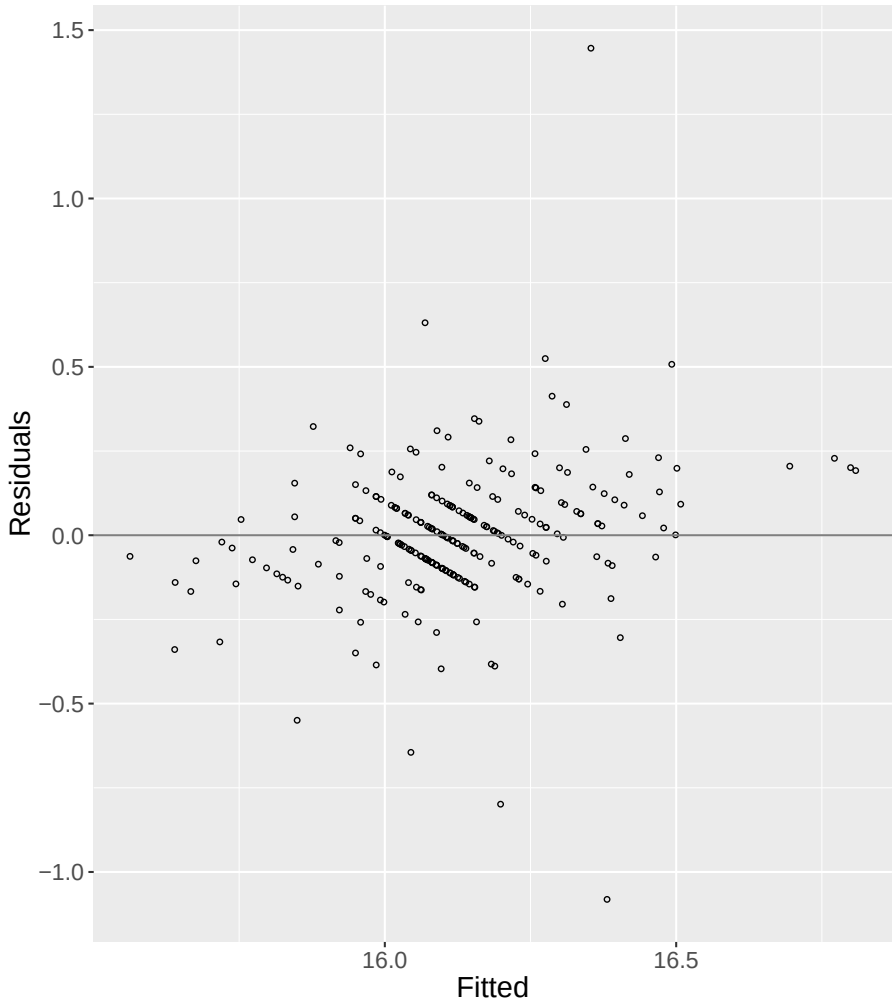

Figure 7.2: Residuals vs fitted values for the voltage data

The model omitting the `Setstat:Teststat` term returns a REML log-likelihood of 203.242 - the same as the REML log-likelihood for the previous model:

```
> asreml.options(gammaPar = TRUE)
> voltage2.asr <- asreml(voltage ~ 1, random = ~Setstat + Setstat:Regulatr + Teststat,
    residual = ~idv(units), data = voltage)
```

A summary of the variance components for this model is:

```
> summary(voltage2.asr)$varcomp
```

```
                  component   std.error   z.ratio bound %ch
Teststat         0.003287145 0.003337467 0.9849221     P   0
Setstat          0.011937344 0.008812595 1.3545776     P   0
Setstat:Regulatr 0.030777971 0.008452070 3.6414712     P   0
units!units      0.051141732          NA        NA     F   0
units!R          0.051141732 0.005260980 9.7209508     P   0
```

The column labelled `z.ratio` is calculated to give a guide as to the significance of the variance components. The statistic is simply the REML estimate of the variance component divided by the square root of the diagonal element (for each component) of the inverse of the average information matrix. The diagonal elements of the expected information matrix are the asymptotic variances of the REML estimates of the variance parameters. These statistics cannot be used to test the null hypothesis that the variance component is zero. The conclusions using this crude measure are inconsistent with the conclusions obtained from the REML log-likelihood ratio (Table 7.3).

Table 7.3: REML log-likelihood ratio test for each variance component in the voltage data

term	log-likelihood	$-2\times$ difference	P-value
`Setstat`	200.31	5.864	.0077
`Setstat:Regulatr`	184.15	38.19	.0000
`Teststat`	199.71	7.064	.0039

7.5 Balanced repeated measures

The data for this example comes from an experiment conducted at Rothamstead Experimental Station, UK, by J. Lamptey. It consists of a total of 5 measurements of height (cm) taken on 14 plants. 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. The data are illustrated in Figure 7.3.

The following illustrates several repeated measures analyses. For some of these it is convenient to arrange the data in a multivariate form, with 7 columns containing the plant number, treatment identification and the 5 heights, respectively, while for other analyses, in particular power models, it is necessary to expand the data frame in a relational sense so that the response, response names and a variate for the time of measurement occupy one column each.

The data frame grass is in *multivariate* form:

```
grass
```

```
   Tmt Plant   y1   y3    y5    y7   y10
1  MAV     1 21.0 39.7  47.0  53.0  55.0
2  MAV     2 32.0 59.5  63.5  65.0  67.6
3  MAV     3 35.5 54.6  58.0  61.5  61.5
4  MAV     4 33.5 41.0  48.0  57.0  58.0
5  MAV     5 31.5 45.3  62.0 104.0 104.0
```

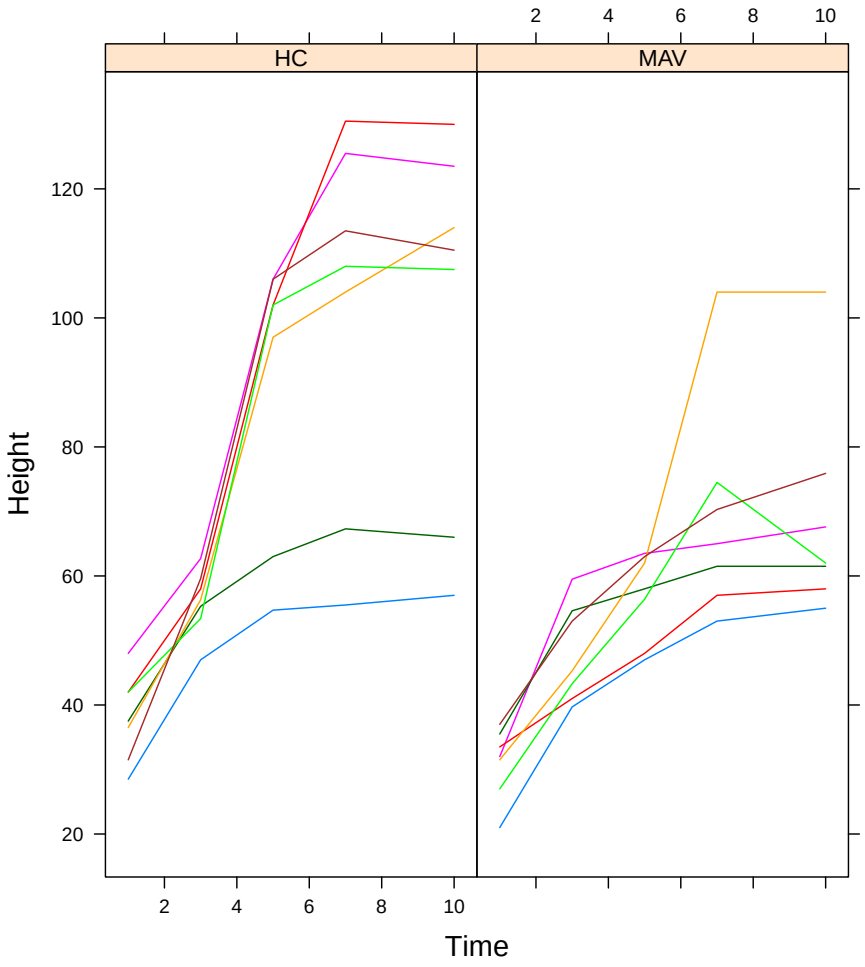

Figure 7.3: Trellis plot of plant height for each of 14 plants

```
6   MAV     6 27.0 43.3  56.4  74.5  62.0
7   MAV     7 37.0 53.0  63.0  70.3  75.9
8    HC     8 28.5 47.0  54.7  55.5  57.0
9    HC     9 48.0 62.7 106.0 125.5 123.5
10   HC    10 37.5 55.3  63.0  67.3  66.0
11   HC    11 42.0 58.0 102.0 130.5 130.0
12   HC    12 36.5 56.3  97.0 104.0 114.0
13   HC    13 42.0 53.4 102.0 108.0 107.5
14   HC    14 31.5 59.6 106.0 113.5 110.5
```

while `grassUV` is in *univariate* form:

```
grassUV
```

```
   Tmt Plant Time HeightID     y
1  MAV     1    1       y1  21.0
2  MAV     1    3       y3  39.7
```

```
3  MAV     1    5       y5  47.0
4  MAV     1    7       y7  53.0
5  MAV     1   10      y10  55.0
6  MAV     2    1       y1  32.0
7  MAV     2    3       y3  59.5
8  MAV     2    5       y5  63.5
9  MAV     2    7       y7  65.0
10 MAV     2   10      y10  67.6
11 MAV     3    1       y1  35.5
12 MAV     3    3       y3  54.6
13 MAV     3    5       y5  58.0
14 MAV     3    7       y7  61.5
15 MAV     3   10      y10  61.5
16 MAV     4    1       y1  33.5
17 MAV     4    3       y3  41.0
18 MAV     4    5       y5  48.0
19 MAV     4    7       y7  57.0
20 MAV     4   10      y10  58.0
21 MAV     5    1       y1  31.5
22 MAV     5    3       y3  45.3
23 MAV     5    5       y5  62.0
24 MAV     5    7       y7 104.0
25 MAV     5   10      y10 104.0
26 MAV     6    1       y1  27.0
27 MAV     6    3       y3  43.3
28 MAV     6    5       y5  56.4
29 MAV     6    7       y7  74.5
30 MAV     6   10      y10  62.0
31 MAV     7    1       y1  37.0
32 MAV     7    3       y3  53.0
33 MAV     7    5       y5  63.0
34 MAV     7    7       y7  70.3
35 MAV     7   10      y10  75.9
36  HC     8    1       y1  28.5
37  HC     8    3       y3  47.0
38  HC     8    5       y5  54.7
39  HC     8    7       y7  55.5
40  HC     8   10      y10  57.0
41  HC     9    1       y1  48.0
42  HC     9    3       y3  62.7
43  HC     9    5       y5 106.0
44  HC     9    7       y7 125.5
45  HC     9   10      y10 123.5
46  HC    10    1       y1  37.5
47  HC    10    3       y3  55.3
48  HC    10    5       y5  63.0
49  HC    10    7       y7  67.3
50  HC    10   10      y10  66.0
51  HC    11    1       y1  42.0
52  HC    11    3       y3  58.0
53  HC    11    5       y5 102.0
54  HC    11    7       y7 130.5
55  HC    11   10      y10 130.0
56  HC    12    1       y1  36.5
57  HC    12    3       y3  56.3
```

```
58  HC    12    5       y5  97.0
59  HC    12    7       y7 104.0
60  HC    12   10      y10 114.0
61  HC    13    1       y1  42.0
62  HC    13    3       y3  53.4
63  HC    13    5       y5 102.0
64  HC    13    7       y7 108.0
65  HC    13   10      y10 107.5
66  HC    14    1       y1  31.5
67  HC    14    3       y3  59.6
68  HC    14    5       y5 106.0
69  HC    14    7       y7 113.5
70  HC    14   10      y10 110.5
```

The focus is on modelling the error variance for the data. Specifically we fit the multivariate regression model given by

$$\boldsymbol{Y} = \boldsymbol{DT} + \boldsymbol{E} \tag{7.1}$$

where $\boldsymbol{Y}^{14\times5}$ is the matrix of heights, $\boldsymbol{D}^{14\times2}$ is the design matrix, $\boldsymbol{T}^{2\times5}$ is the matrix of fixed effects and $\boldsymbol{E}^{14\times5}$ is the matrix of errors. The heights taken on the same plants will be correlated and so we assume that

$$\text{var}\left(vec(\boldsymbol{E})\right) = \boldsymbol{I}_{14} \otimes \boldsymbol{\Sigma} \tag{7.2}$$

where $\boldsymbol{\Sigma}^{5\times5}$ is a symmetric positive definite matrix.

The variance models used for $\boldsymbol{\Sigma}$ are summarised in Table 7.4. They represent some commonly used models for the analysis of repeated measures data (Wolfinger; 1996). The variance models are fitted by specifying the appropriate special function in the asreml() call.

The sequence of models given below illustrate some important issues regarding the sort order of the data. In a standard multivariate analysis (data frame `grass`) the response is specified as a matrix and ASReml-R automatically expands the data frame internally to a univariate form in the order **trait nested within units**. The factor `units` is created before this expansion. The data frame `grassUV` has been expanded outside ASReml-R in the same order, that is **trait nested within experimental units**. In this case ASReml-R cannot sensibly create a correct `units` factor so a factor defining the experimental units must already exist - in this case the factor `Plant` can be used. Note that the sort order of the data must correspond to the order of appearance of the factors in the `residual` formula that defines the experimental units. In the case of the one dimensional power model, the data must be sorted in the order returned by `unique(x)` where `x` is the column in the data frame containing the distances. In this case ASReml-R checks the sort order and reports an error if incorrect.

Uniform

```
asreml.options(gammaPar = TRUE)
grass.asr <- asreml(cbind(y1, y3, y5, y7, y10) ~ trait + Tmt + trait:Tmt, random =
    ~idv(units), residual = ~id(units):idv(trait), data = grass)
```

Power

```
asreml.options(gammaPar = TRUE)
grass2.asr <- asreml(y ~ Tmt + Time + Tmt:Time, residual = ~id(Plant):expv(Time), data =
    grassUV)
```

Heterogeneous power

```
grass3.asr <- asreml(y ~ Tmt + Time + Tmt:Time, residual = ~id(Plant):exph(Time), data =
    grassUV)
```

Antedependence

```
grass4.asr <- asreml(cbind(y1, y3, y5, y7, y10) ~ trait + Tmt + trait:Tmt, residual =
    ~id(units):ante(trait, 1), data = grass)
```

In this case the residual specifies a variance matrix so the sigma parameterization is used for estimation, see Section 2.1.1. The default for multivariate analyses is to use the sigma parameterization.

Unstructured

```
grass5.asr <- asreml(cbind(y1, y3, y5, y7, y10) ~ trait + Tmt + trait:Tmt, residual =
    ~id(units):us(trait), data = grass)
```

Table 7.4: Summary of variance models fitted to the plant data

model	number of parameters	REML log-likelihood	BIC
uniform	2	-196.88	401.95
power	2	-182.98	374.15
heterogeneous power	6	-171.50	367.57
antedependence (order 1)	9	-160.37	357.51
unstructured	15	-158.04	377.50

The split plot in time model can be fitted four ways:

1. by fitting a random `units` term plus an independent residual using the *multivariate* data frame.

2. by specifying a `cor()` variance model for the *R*-structure, again using the *multivariate* data frame;

```
asreml.options(gammaPar = TRUE)
grass1.asr <- asreml(cbind(y1, y3, y5, y7, y10) ~ trait + Tmt + trait:Tmt, residual =
    ~id(units):corv(trait), data = grass)
```

As this residual is specified as a correlation matrix the model is fitted on the gamma scale. The residual variance is denoted as `S2` in the convergence trace.

3. by fitting `Plant` as a random term plus an independent residual (`Time:Plant`) using the *univariate* data frame,

4. by specifying a `cor()` variance model for the `Time:Plant` residual term using the *univariate* data.

1. and **3.** are equivalent as are **2.** and **4.** The two forms for Σ are given by

$$\boldsymbol{\Sigma} = \sigma_1^2 \boldsymbol{J} + \sigma_2^2 \boldsymbol{I}, \qquad \texttt{units} \tag{7.3}$$

$$\boldsymbol{\Sigma} = \sigma_e^2 \boldsymbol{I} + \sigma_e^2 \rho (\boldsymbol{J} - \boldsymbol{I}), \qquad \texttt{cor()} \tag{7.4}$$

It follows that

$$\begin{aligned} \sigma_e^2 &= \sigma_1^2 + \sigma_2^2 \\ \rho &= \frac{\sigma_1^2}{\sigma_1^2 + \sigma_2^2} \end{aligned} \tag{7.5}$$

Summaries of the outputs from **1.** and **2.** (the asreml() calls labelled **Uniform** and **Correlation**, respectively) are given below. The REML log-likelihood is the same for both models and it is easy to verify that the REML estimates of the variance parameters satisfy (7.5).

```
summary(grass.asr)$loglik
```

```
[1] -196.8768
```

```
summary(grass.asr)$varcomp
```

```
                 component std.error  z.ratio bound %ch
units!units       159.8161  75.74758 2.109851     P   0
units:trait!trait 126.4946        NA       NA     F   0
units:trait!R     126.4946  25.82064 4.898972     P   0
```

```
summary(grass1.asr)$loglik
```

```
[1] -196.8768
```

```
summary(grass1.asr)$varcomp
```

```
                       component  std.error  z.ratio bound %ch
units:trait!R         286.3088841 78.3441796 3.654501     P   0
units:trait!trait!cor   0.5581911  0.1303821 4.281196     U   0
units:trait!trait!var 286.3088841         NA       NA     F   0
```

A more plausible model for repeated measures data would allow the correlations to decrease as the lag increases. The simplest model that accommodates this is the first order autoregressive model. However, since the heights are not measured at equally spaced time points we use the exp() power model. The correlation function is given by:

$$\rho(u) = \phi^u$$

where u is the time lag is weeks. The variance parameters from this model are:

```
summary(grass2.asr)$varcomp
```

```
                      component   std.error   z.ratio bound %ch
Plant:Time!R        300.9211889 96.38923337  3.121938     P   0
Plant:Time!Time!pow   0.9190407  0.03120443 29.452250     U   0
Plant:Time!Time!var 300.9211889          NA        NA     F   0
```

When fitting such models be careful to ensure the scale of the defining variate, `Time` here, does not result in an estimate of ϕ too close to 1. For example, use of days in this example would result in an estimate for ϕ of about 0.993.

```
plot(grass2.asr, formula = resid(.) ~ Plant | Time, fun = "xyplot")
```

creates a trend plot (Figure 7.4) of residuals against the factors that index the experimental units.

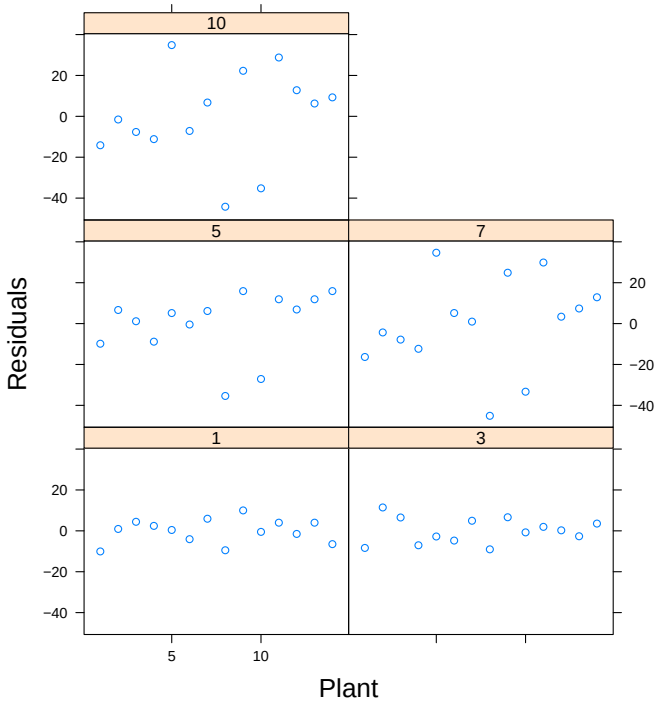

Figure 7.4: `residual` ∼ `Plant | Time` for the `exp()` variance model for the plant data

The residual plot suggests increasing variance over time. This can be modelled via the `exph()` variance function, which models $\boldsymbol{\Sigma}$ by

$$\boldsymbol{\Sigma} = \boldsymbol{D}^{0.5}\boldsymbol{C}\boldsymbol{D}^{0.5}$$

where $\boldsymbol{D}$ is a diagonal matrix of variances and $\boldsymbol{C}$ is a correlation matrix with elements given by $c_{ij} = \phi^{|t_i - t_j|}$. Parameter estimates for the **Heterogeneous power** model are:

```
summary(grass3.asr)$varcomp
```

```
                      component    std.error   z.ratio bound %ch
Plant:Time!R          1.0000000           NA        NA     F 0.0
Plant:Time!Time!pow   0.9071081   0.04128149 21.973724     U 0.0
Plant:Time!Time_1    61.1918427  29.01081243  2.109277     P 0.6
Plant:Time!Time_3    72.6870993  36.24111494  2.005653     P 0.6
Plant:Time!Time_5   309.9358176 140.13966884  2.211621     P 0.4
Plant:Time!Time_7   437.5366349 174.82222681  2.502752     P 0.5
Plant:Time!Time_10  383.2658238 140.53497675  2.727192     P 0.3
```

Note that ASReml-R fixes the scale parameter to 1 to ensure that the elements of $\boldsymbol{D}$ are identifiable. The final two models considered are the antedependence model of order 1 and the unstructured model. Both require as starting values the generates starting gammas of 0.15 for variances and 0.10 for covariances and scales these by 1/2 of the simple variance of the response. This is adequate in many cases (including this example) but we would generally recommend using the REML estimate of $\boldsymbol{\Sigma}$ from a previous model. For example, suitable starting values could be generated from the heterogeneous power model (`grass3.asr`) by:

```
r <- matrix(resid(grass3.asr), nrow = 14, byrow = T)
vcov <- (t(r) %*% r)/12
```

where 12 is the degrees of freedom in this case.

The antedependence form models $\boldsymbol{\Sigma}$ by the inverse cholesky decomposition

$$\boldsymbol{\Sigma} = \boldsymbol{U}\boldsymbol{D}\boldsymbol{U}'$$

where $\boldsymbol{D}$ is a diagonal matrix and $\boldsymbol{U}$ is a unit upper triangular matrix. For an antedependence model of order q, then $l_{ij} = 0$ for $j > i + q - 1$. The antedependence model of order 1 has 9 parameters for these data, 5 in $\boldsymbol{D}$ and 4 in $\boldsymbol{U}$. The call using the default starting values is shown above.

The antedependence parameter estimates are given below and appear successively for each time, that is, the element of $\boldsymbol{D}$ and then the row of $\boldsymbol{U}$:

```
summary(grass4.asr)$varcomp
```

```
                            component   std.error    z.ratio bound %ch
units:trait!R             1.000000000          NA         NA     F 0.0
units:trait!trait_y1:y1     0.026866609 0.011023605   2.437189     U 0.0
units:trait!trait_y3:y1    -0.628374025 0.246035667  -2.553996     U 0.0
units:trait!trait_y3:y3     0.037282432 0.015467375   2.410392     U 0.0
units:trait!trait_y5:y3    -1.491096654 0.586492845  -2.542395     U 0.1
units:trait!trait_y5:y5     0.005996185 0.002467878   2.429692     U 0.0
units:trait!trait_y7:y5    -1.280576604 0.206798510  -6.192388     U 0.0
units:trait!trait_y7:y7     0.007896552 0.003232983   2.442497     U 0.0
units:trait!trait_y10:y7   -0.967807268 0.062828991 -15.403833     U 0.0
units:trait!trait_y10:y10   0.039063461 0.015947580   2.449491     U 0.0
```

Finally, the estimated components for the unstructured model using default starting values.

```
summary(grass5.asr)$varcomp
```

```
                          component std.error  z.ratio bound %ch
units:trait!R               1.00000        NA       NA     F   0
units:trait!trait_y1:y1    37.22619  15.19746 2.449501     P   0
units:trait!trait_y3:y1    23.39345  13.20622 1.771397     P   0
units:trait!trait_y3:y3    41.51952  16.95020 2.449500     P   0
units:trait!trait_y5:y1    51.65238  32.03385 1.612431     P   0
units:trait!trait_y5:y3    61.91690  34.87146 1.775575     P   0
units:trait!trait_y5:y5   259.12143 105.78573 2.449493     P   0
units:trait!trait_y7:y1    70.81131  46.13796 1.534773     P   0
units:trait!trait_y7:y3    57.61452  46.74179 1.232613     P   0
units:trait!trait_y7:y5   331.80679 145.20148 2.285147     P   0
units:trait!trait_y7:y7   551.50690 225.15083 2.449500     P   0
units:trait!trait_y10:y1   73.78571  46.21260 1.596658     P   0
units:trait!trait_y10:y3   62.56905  46.92685 1.333331     P   0
units:trait!trait_y10:y5  330.85060 144.32316 2.292429     P   0
units:trait!trait_y10:y7  533.75583 220.58712 2.419705     P   0
units:trait!trait_y10:y10 542.17548 221.34133 2.449499     P   0
```

Table 7.5: Summary of Wald statistics for fixed effects for the models fitted to the plant data

model	`Tmt` (df=1)	`trait:Tmt` (df=4)
uniform	9.42	20.40
power	6.85	24.53
heterogeneous power	0.00	19.28
antedependence (order 1)	4.19	15.63
unstructured	1.72	17.86

The antedependence model of order 1 is clearly the more parsimonious model (Table 7.4). There is a surprising level of discrepancy between models for the Wald tests (Table 7.5). The main effect of treatment is significant for the uniform, power and antedependence models.

7.6 Spatial analysis of a field experiment

This section illustrates spatial and incomplete block analyses of a field experiment using ASReml-R. There has been a large amount of interest in developing techniques for the analysis of spatial data both in the context of field experiments and geostatistical data (Cullis and Gleeson; 1991; Cressie; 1991; Gilmour et al.; 1997, for example). This example illustrates the analysis of *so-called* regular spatial data, in which the data is observed on a lattice or regular grid. This is typical of most small plot designed field experiments. Spatial data is often irregularly spaced, either by design or because of the observational nature of the study. The techniques we present in the following can be extended for the analysis of irregularly spaced spatial data, though, larger spatial data-sets may be computationally challenging, depending on the degree of irregularity or models fitted.

The data appears in Gilmour et al. (1995) and is from a field experiment designed to compare the performance of 25 varieties of barley. The experiment was conducted at Slate Hall Farm, UK, in 1976 and was designed as a balanced lattice square with 6 replicates laid out in a 10×15 rectangular grid. Table 7.6 shows the layout of the experiment and the coding of the replicates and lattice blocks. The columns in the data frame are:

7.6 Spatial analysis of a field experiment

```
shf <- asreml.read.table("../examples/shf.csv", header = T, sep = ",")
names(shf)
```

```
[1] "Rep"     "RowBlk"  "ColBlk"  "Row"     "Column"  "Variety" "yield"
```

Lattice block numbering is typically coded *within* replicates, however, in this example the lattice row and column blocks were both numbered from 1 to 30. The terms in the linear model are therefore simply `RowBlk` and `ColBlk`. The factors `Row` and `Column` indicate the spatial layout of the plots.

Table 7.6: Field layout of Slate Hall Farm experiment

Column - Replicate levels															
Row	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	1	1	1	1	1	2	2	2	2	2	3	3	3	3	3
2	1	1	1	1	1	2	2	2	2	2	3	3	3	3	3
3	1	1	1	1	1	2	2	2	2	2	3	3	3	3	3
4	1	1	1	1	1	2	2	2	2	2	3	3	3	3	3
5	1	1	1	1	1	2	2	2	2	2	3	3	3	3	3
6	4	4	4	4	4	5	5	5	5	5	6	6	6	6	6
7	4	4	4	4	4	5	5	5	5	5	6	6	6	6	6
8	4	4	4	4	4	5	5	5	5	5	6	6	6	6	6
9	4	4	4	4	4	5	5	5	5	5	6	6	6	6	6
10	4	4	4	4	4	5	5	5	5	5	6	6	6	6	6
Column - Rowblk levels															
Row	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	1	1	1	1	1	11	11	11	11	11	21	21	21	21	21
2	2	2	2	2	2	12	12	12	12	12	22	22	22	22	22
3	3	3	3	3	3	13	13	13	13	13	23	23	23	23	23
4	4	4	4	4	4	14	14	14	14	14	24	24	24	24	24
5	5	5	5	5	5	15	15	15	15	15	25	25	25	25	25
6	6	6	6	6	6	16	16	16	16	16	26	26	26	26	26
7	7	7	7	7	7	17	17	17	17	17	27	27	27	27	27
8	8	8	8	8	8	18	18	18	18	18	28	28	28	28	28
9	9	9	9	9	9	19	19	19	19	19	29	29	29	29	29
10	10	10	10	10	10	20	20	20	20	20	30	30	30	30	30
Column - Colblk levels															
Row	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
2	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
3	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
4	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
5	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
6	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
7	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
8	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
9	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
10	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30

Three models are considered: two spatial and the traditional incomplete block for comparative purposes. In the first model we fit a separable first order autoregressive process to the variance structure of the plot errors. Gilmour et al. (1997) suggest this is often a useful model to commence the spatial modelling process. The form of the variance matrix for the plot errors ($\boldsymbol{R}$-structure) is given by

$$\sigma^2\boldsymbol{\Sigma} = \sigma^2(\boldsymbol{\Sigma}_c \otimes \boldsymbol{\Sigma}_r) \tag{7.6}$$

where $\boldsymbol{\Sigma}_c$ and $\boldsymbol{\Sigma}_r$ are 15×15 and 10×10 matrix functions of the column (ϕ_c) and row (ϕ_r) autoregressive parameters respectively.

7.6 Spatial analysis of a field experiment

Gilmour et al. (1997) recommend revision of the current spatial model based on the use of diagnostics such as the sample variogram of the residuals. This diagnostic and a summary of row and column residual trends are produced by the `varioGram()` and `plot()` methods.

The separable autoregressive error model is fitted by:

```
asreml.options(gammaPar = TRUE)
barley1.asr <- asreml(yield ~ Variety, residual = ~ar1v(Row):ar1(Column), data = shf)
```

The REML log-likelihood, random components and Wald statistics from the fit are:

```
barley1.asr$loglik
```

```
[1] -700.3226
```

```
summary(barley1.asr)$varcomp
```

```
                          component    std.error   z.ratio bound %ch
Row:Column!R          3.871303e+04 7.737983e+03  5.002987     P 0.0
Row:Column!Row!cor    4.583949e-01 8.261585e-02  5.548510     U 0.2
Row:Column!Row!var    3.871303e+04           NA        NA     F 0.0
Row:Column!Column!cor 6.837847e-01 6.330001e-02 10.802284     U 0.0
```

```
wald(barley1.asr)
```

```
Wald tests for fixed effects.
Response: yield
Terms added sequentially; adjusted for those above.

             Df Sum of Sq Wald statistic Pr(Chisq)
(Intercept)   1  33049749         853.71 < 2.2e-16 ***
Variety      24  12102335         312.62 < 2.2e-16 ***
residual (MS)        38713
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
plot(varioGram(barley1.asr))
```

plots the sample variogram shown in Figure 7.5.

The iterative sequence has converged to *column* and *row* correlation parameters of 0.68378 and 0.45851, respectively. The plot size and orientation is not known and so it is not possible to ascertain whether these values are spatially sensible. It is generally found that the closer the plot centroids, the higher the spatial correlation. This is not always the case and if the highest between plot correlation relates to the larger spatial distance then this may suggest the presence of extraneous variation (Gilmour et al.; 1997, for example). The plot of the sample variogram of the residuals is not trimmed and, ignoring the unreliable contribution from extreme lags, appears in reasonable agreement with the model.

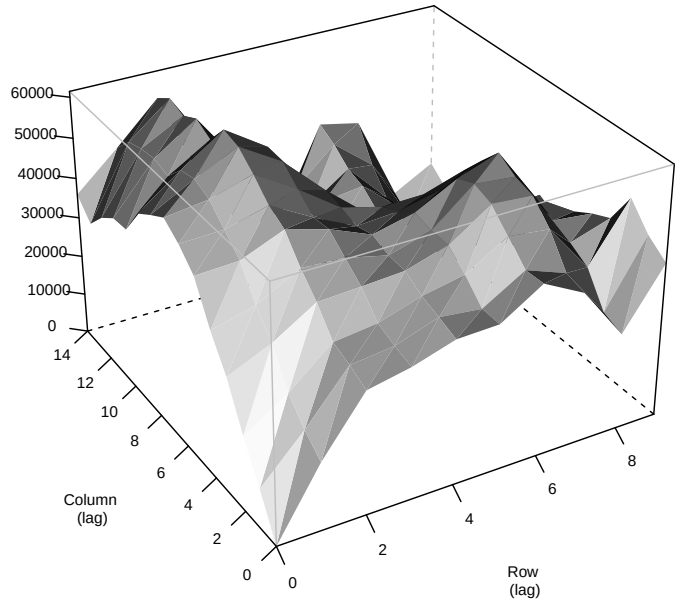

Figure 7.5: Sample variogram of the AR1×AR1 model for the Slate Hall data

An extension to this model includes a measurement error or *nugget effect* term:

```
asreml.options(gammaPar = TRUE)
barley2.asr <- asreml(yield ~ Variety, random = ~idv(units), residual =
    ~ar1v(Row):ar1(Column), data = shf)
```

That is, the variance model for the plot errors is now given by

$$\sigma^2\boldsymbol{\Sigma} = \sigma^2(\boldsymbol{\Sigma}_c \otimes \boldsymbol{\Sigma}_r) + \psi \boldsymbol{I}_{150} \tag{7.7}$$

where ψ is the ratio of nugget variance to error variance (σ^2). The results show a significant improvement in the REML log-likelihood with the inclusion of the nugget effect (Table 7.7).

```
barley2.asr$loglik
```

```
[1] -696.8227
```

```
summary(barley2.asr)$varcomp
```

```
                         component    std.error   z.ratio bound %ch
units!units           4.859729e+03 1.787790e+03  2.718289     P   0
Row:Column!R          4.577444e+04 1.667775e+04  2.744641     P   0
Row:Column!Row!cor    6.826634e-01 1.022774e-01  6.674629     U   0
Row:Column!Row!var    4.577444e+04           NA        NA     F   0
Row:Column!Column!cor 8.437903e-01 6.847152e-02 12.323230     U   0
```

```
wald(barley2.asr)
```

```
Wald tests for fixed effects.
Response: yield
Terms added sequentially; adjusted for those above.

             Df Sum of Sq Wald statistic Pr(Chisq)
(Intercept)   1  11911581         260.22 < 2.2e-16 ***
Variety      24  11235342         245.45 < 2.2e-16 ***
residual (MS)        45774
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The incomplete block analysis (with recovery of inter-block information) is:

```
barley3.asr <- asreml(yield ~ Variety, random = ~Rep + RowBlk + ColBlk, data = shf)
```

```
barley3.asr$loglik
```

```
[1] -707.7857
```

```
summary(barley3.asr)$varcomp
```

```
        component std.error  z.ratio bound %ch
Rep      4262.685  6876.542 0.619888     P 0.4
RowBlk  15596.231  5089.914 3.064144     P 0.0
ColBlk  14812.688  4866.270 3.043951     P 0.0
units!R  8062.414  1340.612 6.013979     P 0.0
```

```
wald(barley3.asr)
```

```
Wald tests for fixed effects.
Response: yield
Terms added sequentially; adjusted for those above.

             Df Sum of Sq Wald statistic Pr(Chisq)
(Intercept)   1   9819711        1217.96 < 2.2e-16 ***
Variety      24   1711182         212.24 < 2.2e-16 ***
residual (MS)         8062
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

This variance model is not competitive with the preceding spatial models. The models can be formally compared using the BIC values, for example.

The Wald statistics for the spatial models are greater than that for the incomplete block analysis (Table 7.7). We note that the Wald statistic for the spatial model including the nugget effect is smaller than that for the AR1×AR1 model.

Table 7.7: Summary of models fitted to the Slate Hall data

model	REML log-likelihood	parameters	Wald statistic	sed
AR1×AR1	-700.32	3	312.82	59.0
AR1×AR1 + units	-696.82	4	245.49	60.5
incomplete block	-707.79	4	212.26	62.0

Finally, we predict `Variety` means for each model using the predict() method. Only the first five and final three means are reproduced here. The overall SED is the square root of the average variance of difference between the variety means. The two spatial analyses have a range of SEDs which may be obtained in matrix form from the `sed` argument of predict(). Note that all variety comparisons have the same SED for the balanced lattice square analysis.

```
barley1.pv <- predict(barley1.asr, classify = "Variety")
```

```
barley1.pv$pvals
```

```
Notes:
- The predictions are obtained by averaging across the hypertable
  calculated from model terms constructed solely from factors in
  the averaging and classify sets.
- Use 'average' to move ignored factors into the averaging set.
```

```
   Variety predicted.value std.error    status
1        1        1257.980  64.61807 Estimable
2        2        1501.443  64.98191 Estimable
3        3        1404.987  64.62963 Estimable
4        4        1412.569  64.90628 Estimable
5        5        1514.480  65.59244 Estimable
6        6        1553.458  64.14977 Estimable
7        7        1379.008  64.15830 Estimable
8        8        1475.891  64.45772 Estimable
9        9        1275.473  64.30651 Estimable
10      10        1212.992  63.94961 Estimable
11      11        1342.502  64.50710 Estimable
12      12        1455.240  64.10906 Estimable
13      13        1658.459  63.23078 Estimable
14      14        1298.247  65.26290 Estimable
15      15        1455.522  64.03194 Estimable
16      16        1296.928  64.49421 Estimable
17      17        1499.193  63.17865 Estimable
18      18        1512.119  63.85935 Estimable
19      19        1653.810  64.39608 Estimable
20      20        1674.083  63.97367 Estimable
```

```
21       21         1517.595  64.70899 Estimable
22       22         1605.046  64.38958 Estimable
23       23         1311.488  64.07644 Estimable
24       24         1586.784  64.70404 Estimable
25       25         1592.020  63.59370 Estimable
```

```
barley1.pv$avsed
```

```
 overall
59.05228
```

```
barley2.pv <- predict(barley2.asr, classify = "Variety")
```

```
barley2.pv$pvals
```

```
Notes:
- The predictions are obtained by averaging across the hypertable
  calculated from model terms constructed solely from factors in
  the averaging and classify sets.
- Use 'average' to move ignored factors into the averaging set.
- The ignored set: units
- Variety is included in this prediction
- (Intercept) is included in this prediction
- units is ignored in this prediction
```

```
   Variety predicted.value std.error    status
1        1         1245.582  97.87602 Estimable
2        2         1516.234  97.86415 Estimable
3        3         1403.985  98.25679 Estimable
4        4         1404.918  98.00437 Estimable
5        5         1471.612  98.37758 Estimable
6        6         1521.938  97.98808 Estimable
7        7         1372.820  98.09363 Estimable
8        8         1453.017  98.61442 Estimable
9        9         1262.331  97.96488 Estimable
10      10         1195.635  98.42021 Estimable
11      11         1328.915  98.22051 Estimable
12      12         1441.677  98.40365 Estimable
13      13         1624.213  98.18592 Estimable
14      14         1299.313  98.18378 Estimable
15      15         1469.594  98.21255 Estimable
16      16         1287.519  97.80526 Estimable
17      17         1492.885  97.96819 Estimable
18      18         1527.493  97.94485 Estimable
19      19         1649.209  98.05410 Estimable
20      20         1646.049  98.04581 Estimable
21      21         1514.751  98.15831 Estimable
22      22         1608.953  98.23694 Estimable
23      23         1316.874  98.05723 Estimable
24      24         1557.522  98.14424 Estimable
25      25         1573.888  97.99743 Estimable
```

```
barley2.pv$avsed
```

```
 overall
60.51084
```

```
barley3.pv <- predict(barley3.asr, classify = "Variety")
```

```
barley3.pv$pvals
```

```
Notes:
- The predictions are obtained by averaging across the hypertable
  calculated from model terms constructed solely from factors in
  the averaging and classify sets.
- Use 'average' to move ignored factors into the averaging set.
- The ignored set: Rep,RowBlk,ColBlk
```

	Variety	predicted.value	std.error	status
1	1	1283.587	60.1994	Estimable
2	2	1549.013	60.1994	Estimable
3	3	1420.931	60.1994	Estimable
4	4	1451.855	60.1994	Estimable
5	5	1533.275	60.1994	Estimable
6	6	1527.407	60.1994	Estimable
7	7	1400.728	60.1994	Estimable
8	8	1457.374	60.1994	Estimable
9	9	1298.859	60.1994	Estimable
10	10	1193.224	60.1994	Estimable
11	11	1327.245	60.1994	Estimable
12	12	1483.789	60.1994	Estimable
13	13	1619.043	60.1994	Estimable
14	14	1326.645	60.1994	Estimable
15	15	1498.011	60.1994	Estimable
16	16	1346.148	60.1994	Estimable
17	17	1498.166	60.1994	Estimable
18	18	1592.177	60.1994	Estimable
19	19	1669.551	60.1994	Estimable
20	20	1639.946	60.1994	Estimable
21	21	1493.437	60.1994	Estimable
22	22	1644.381	60.1994	Estimable
23	23	1329.109	60.1994	Estimable
24	24	1546.470	60.1994	Estimable
25	25	1630.629	60.1994	Estimable

```
barley3.pv$avsed
```

```
 overall
62.01934
```

7.7 Unreplicated early generation variety trial

This example is a further illustration of the approach to the analysis of field trials presented in the previous section. The data are from an unreplicated field experiment conducted at Tullibigeal in south-western NSW. The trial was an S1 (early stage) wheat variety evaluation trial and consisted of 525 test lines which were randomly assigned to plots in a 67 row $\times$ 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. The aim of these trials is to identify and retain the top, say 20%, lines for further testing. Cullis et al. (1989) considered the analysis of early generation variety trials and presented a one-dimensional spatial analysis which was an extension of the approach developed by Gleeson and Cullis (1987). The test line effects are assumed random, while the check variety effects are considered fixed. This may not be sensible or justifiable for most trials and can lead to inconsistent comparisons between check varieties and test lines. Given the large amount of replication afforded to check varieties there will be very little shrinkage irrespective of the realised heritability.

In the following we assume that the variety effect (including both check, replicated and unreplicated lines) is random. In addition to a one dimensional analysis we consider three further spatial models for comparison.

```
wheat <- asreml.read.table("../examples/wheat.csv", header = T, sep = ",")
```

```
names(wheat)
```

```
[1] "yield"   "weed"    "Column"  "Row"     "Variety"
```

where `Variety, Row` and `Column` are factors, `yield` is the response variate and `weed` is a covariate. Note that the data frame is sorted as *Column* nested within *Row*.

We begin with a one-dimensional spatial model, which assumes the variance model for the plot effects within columns is described by a first order autoregressive process.

```
asreml.options(gammaPar = TRUE)
wheat1.asr <- asreml(yield ~ weed, random = ~idv(Variety), residual =
    ~ar1v(Row):id(Column), data = wheat)
```

The REML log-likelihood, random components and Wald statistics from the fit are:

```
wheat1.asr$loglik
```

```
[1] -4239.88
```

```
summary(wheat1.asr)$varcomp
```

```
                     component    std.error   z.ratio bound %ch
Variety!Variety    82788.464274 9.217842e+03  8.981328     P 0.0
```

```
Row:Column!R       86289.992260 9.460971e+03  9.120628     P 0.0
Row:Column!Row!cor     0.672284 4.185343e-02 16.062820     U 0.1
Row:Column!Row!var 86289.992260           NA        NA     F 0.0
```

The REML estimate of the autoregressive parameter indicates substantial within column heterogeneity.

A two dimensional spatial model is fitted with:

```
asreml.options(gammaPar = TRUE)
wheat2.asr <- asreml(yield ~ weed, random = ~idv(Variety), residual =
    ~ar1v(Row):ar1(Column), data = wheat)
```

```
wheat2.asr$loglik
```

```
[1] -4233.647
```

```
summary(wheat2.asr)$varcomp
```

```
                         component    std.error   z.ratio bound %ch
Variety!Variety       8.810642e+04 8.885330e+03  9.915941     P 0.0
Row:Column!R          8.309418e+04 9.338134e+03  8.898371     P 0.0
Row:Column!Row!cor    6.853206e-01 4.116549e-02 16.647940     U 0.1
Row:Column!Row!var    8.309418e+04           NA        NA     F 0.0
Row:Column!Column!cor 2.858594e-01 7.390089e-02  3.868146     U 0.1
```

The change in REML log-likelihood is significant ($\chi^2_1 = 12.46, P < 0.001$) with the inclusion of the autoregressive parameter for `Column`. The sample variogram of the residuals for the AR1×AR1 model, Figure 7.6, indicates a linear drift from column 1 to column 10. We include a linear regression coefficient `lin(Column)` in the model to account for this.

```
asreml.options(gammaPar = TRUE)
wheat3.asr <- asreml(yield ~ weed + lin(Column), random = ~idv(Variety), residual =
    ~ar1v(Row):ar1(Column), data = wheat)
```

```
wheat3.asr$loglik
```

```
[1] -4227.13
```

```
summary(wheat3.asr)$varcomp
```

```
                         component    std.error   z.ratio bound %ch
Variety!Variety       8.897908e+04 8.976971e+03  9.911927     P 0.0
Row:Column!R          7.780305e+04 8.852879e+03  8.788446     P 0.0
Row:Column!Row!cor    6.713910e-01 4.288854e-02 15.654321     U 0.1
Row:Column!Row!var    7.780305e+04           NA        NA     F 0.0
Row:Column!Column!cor 2.660564e-01 7.541202e-02  3.528037     U 0.1
```

```
wald(wheat3.asr)
```

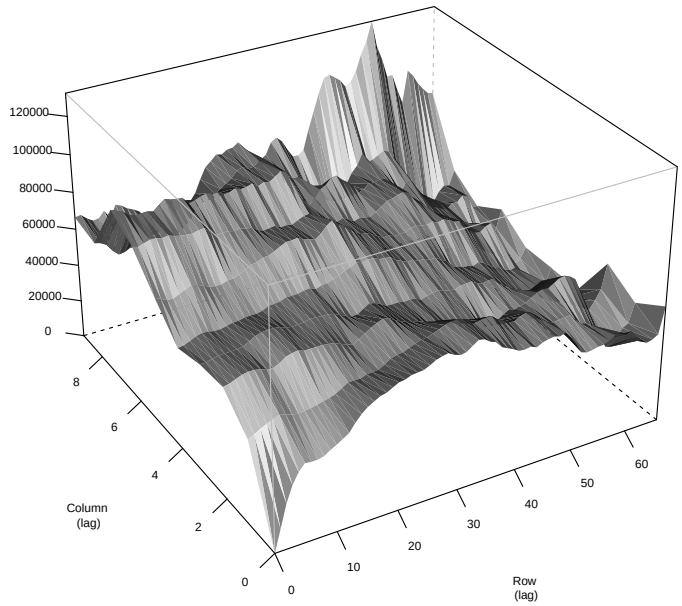

Figure 7.6: Sample variogram of the AR1×AR1 model for the Tullibigeal data

```
Wald tests for fixed effects.
Response: yield
Terms added sequentially; adjusted for those above.

             Df Sum of Sq Wald statistic Pr(Chisq)
(Intercept)   1 551371697         7086.8 < 2.2e-16 ***
weed          1   7156984           92.0 < 2.2e-16 ***
lin(Column)   1    679797            8.7  0.003117 **
residual (MS)       77803
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
wheat3.asr$coefficients$fixed
```

```
                   bu
lin(Column)  -31.1734
weed        -182.7239
(Intercept) 3043.4618
attr(,"terms")
                   tname n
lin(Column) lin(Column) 1
weed               weed 1
(Intercept) (Intercept) 1
```

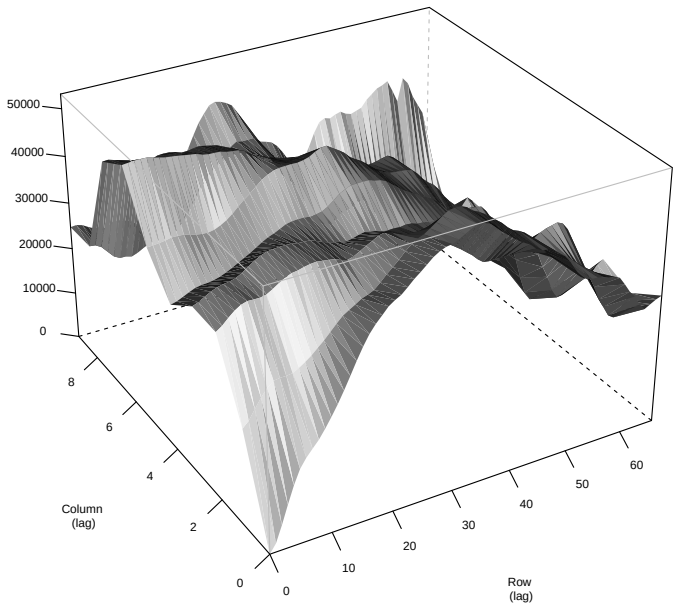

Figure 7.7: Sample variogram of the AR1×AR1 + `lin(column)` model for the Tullibigeal data

The linear regression of column number on yield is significant (Wald statistic = 8.74). The sample variogram (Figure 7.7) seems more satisfactory, though interpretation of variograms is often difficult, particularly for unreplicated trials. This is an issue for further research.

The final model includes a nugget effect:

7.7 Unreplicated early generation variety trial

```
asreml.options(gammaPar = TRUE)
wheat4.asr <- asreml(yield ~ lin(Column), random = ~idv(Variety) + idv(units), sparse =
    ~weed, residual = ~ar1v(Row):ar1(Column), data = wheat)
```

```
wheat4.asr$loglik
```

```
[1] -4221.76
```

```
summary(wheat4.asr)$varcomp
```

```
                          component    std.error   z.ratio bound %ch
Variety!Variety       7.378296e+04 1.041720e+04  7.082798     P   0
units!units           3.044784e+04 8.075220e+03  3.770528     P   0
Row:Column!R          5.472688e+04 1.062781e+04  5.149403     P   0
Row:Column!Row!cor    8.374909e-01 4.487475e-02 18.662854     U   0
Row:Column!Row!var    5.472688e+04           NA        NA     F   0
Row:Column!Column!cor 3.753740e-01 1.152551e-01  3.256898     U   0
```

The increase in REML log-likelihood from adding the `units` term is significant. Predicted variety means can be obtained from this model using

```
wheat4.pv <- predict(wheat4.asr, classify = "Variety:Column", levels = list(Column =
    5.5))
```

Note that the predictions are formed at the average value of `Column`, that is, 5.5.

```
head(wheat4.pv$pvals, 10)
```

```
Notes:
- The predictions are obtained by averaging across the hypertable
  calculated from model terms constructed solely from factors in
  the averaging and classify sets.
- Use 'average' to move ignored factors into the averaging set.
- weed evaluated at average value of 0.459701
- The ignored set: units
- lin(Column) is included in this prediction
- (Intercept) is included in this prediction
- weed is included in this prediction
- units is ignored in this prediction
```

```
   Variety Column predicted.value std.error    status
1        1    5.5        2915.712  179.3117 Estimable
2        2    5.5        2956.275  178.7801 Estimable
3        3    5.5        2871.296  176.9941 Estimable
4        4    5.5        2985.007  178.7450 Estimable
5        5    5.5        2776.809  179.3482 Estimable
6        6    5.5        2799.750  178.9280 Estimable
7        7    5.5        2843.105  178.6194 Estimable
8        8    5.5        3036.838  178.9558 Estimable
9        9    5.5        2921.700  178.9448 Estimable
```

```
10      10    5.5        2836.501  179.2886 Estimable
```

Note that the replicated check lines have lower SEs than the unreplicated test lines. There will also be large differences in SEDs. Rather than obtaining the large table of all SEDs, the prediction could be done in parts if the interest was to to examine the matrix of pairwise prediction errors of check varieties, for example.

```
wheat5.pv <- predict(wheat4.asr, classify = "Variety:Column", levels = list(Variety =
    seq(1, 525), Column = 5.5))
```

```
wheat6.pv <- predict(wheat4.asr, classify = "Variety:Column", levels = list(Variety =
    seq(526, 532), Column = 5.5), sed = T)
```

```
head(wheat6.pv$pvals, 20)
```

```
Notes:
- The predictions are obtained by averaging across the hypertable
  calculated from model terms constructed solely from factors in
  the averaging and classify sets.
- Use 'average' to move ignored factors into the averaging set.
- weed evaluated at average value of 0.459701
- The ignored set: units
- lin(Column) is included in this prediction
- (Intercept) is included in this prediction
- weed is included in this prediction
- units is ignored in this prediction
```

```
  Variety Column predicted.value std.error    status
1     526    5.5        2384.515  44.21822 Estimable
2     527    5.5        2695.601 133.45764 Estimable
3     528    5.5        2725.567 112.26135 Estimable
4     529    5.5        2698.358 103.92510 Estimable
5     530    5.5        3008.925 112.30675 Estimable
6     531    5.5        3018.606 112.27111 Estimable
7     532    5.5        3065.982 112.66740 Estimable
```

```
head(wheat6.pv$avsed, 20)
```

```
      min      mean       max
 98.20978 139.90445 165.97547
```

7.8 Paired Case-Control Study

These data are from an experiment conducted to investigate the tolerance of rice varieties to attack by the larvae of bloodworms. The data have been kindly provided by Dr. Mark Stevens, Yanco Agricultural Institute. A full description of the experiment is given by Stevens et al. (1999). Bloodworms are a significant pest of rice in the Murray and Murrumbidgee irrigation areas and damage can result in poor establishment and substantial yield loss.

7.8 Paired Case-Control Study

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 described by Stevens et al. (1999). 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. Unfortunately the variety concurrence within runs was less than optimal. Eight varieties occurred with only one other variety, 22 with two other varieties and the remaining 14 with three different varieties.

The following sections present an exhaustive analysis of these data using equivalent univariate and multivariate techniques. It is convenient to use two data frames, one for each approach. The univariate data frame has factors `Pair, Run, Variety, Tmt` and variates `rootwt` and `sqrtroot`. The factor `Pair` labels pairs of trays (to which varieties are allocated) and `Tmt` is the two level bloodworm treatment factor (control/treated):

```
rice <- asreml.read.table("../examples/rice.txt", header = T)
```

```
names(rice)
```

```
[1] "Pair"     "rootwt"   "Run"      "sqrtroot" "Tmt"      "Variety"
```

The multivariate data frame contains factors `Variety` and `Run` and variates for root weight and square root of root weight for both the control and exposed treatments (`yc`, `ye`, `syc`, `sye` respectively):

```
riceMV <- asreml.read.table("../examples/riceMV.csv", header = T, sep = ",")
names(riceMV)
```

```
[1] "Pair"    "Run"     "Variety" "yc"      "ye"      "syc"     "sye"
```

A plot of the treated vs the control root area (on the square root scale) for each variety is shown in Figure 7.8. There is a strong dependence between the treated and control root area, which is not surprising. The aim of the experiment was to determine the tolerance of varieties to bloodworms and identify the most tolerant varieties. The definition of tolerance should allow for the fact that varieties differ in their inherent seedling vigour (Figure 7.8). The initial approach was to regress the treated root area against the control root area and define the index of vigour as the residual from this regression. This is clearly inefficient since there is error in both variables. We seek to determine an index of tolerance from the joint analysis of treated and control root area.

Standard analysis

Preliminary analyses indicated variance heterogeneity so that subsequent analyses were conducted on the square root scale. The allocation of bloodworm treatments within varieties and varieties within runs defines a nested block structure of the form:

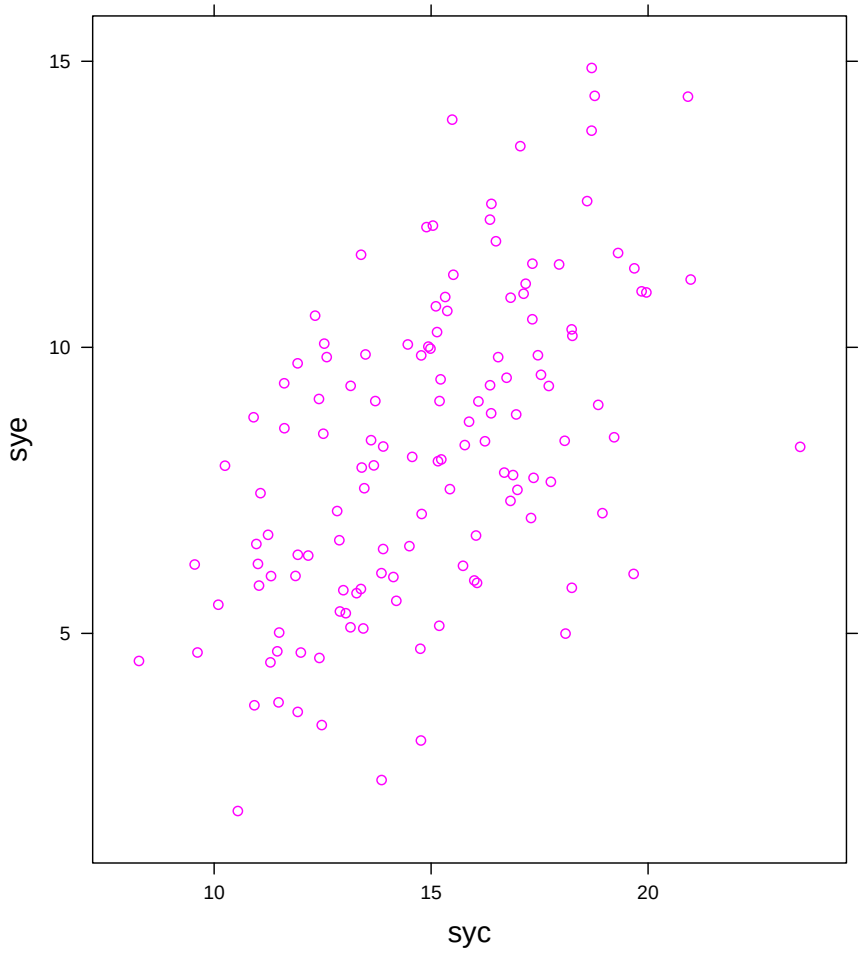

Figure 7.8: Rice bloodworm data: Plot of square root of root weight for treated versus control

```
Run/Variety/Tmt = Run + Run:Variety + Run:Variety:Tmt
= Run + Pair + Pair:Tmt
= Run + Run:Variety + units
```

There is an additional blocking term, however, due to the fact that the bloodworms within a run are derived from the same batch of larvae whereas between runs the bloodworms come from different sources. This defines a block structure of the form:

```
Run/Tmt/Variety = Run + Run:Tmt + Run:Tmt:Variety
= Run + Run:Tmt + Pair:Tmt
```

Combining the two provides the full block structure for the design:

```
Run + Run:Variety + Run:Tmt + Run:Tmt:Variety
= Run + Run:Variety + Run:Tmt + units
= Run + Pair + Run:Tmt + Pair:Tmt
```

7.8 Paired Case-Control Study

In line with the aims of the experiment the treatment structure comprises variety and treatment main effects and treatment by variety interactions.

In the traditional approach the terms in the block structure are regarded as random and the treatment terms as fixed. The choice of treatment terms as fixed or random depends largely on the aims of the experiment. The aim of this example is to select the *best* varieties. The definition of best is somewhat more complex since it does not involve the single trait `sqrt(rootwt)` but rather two traits, namely `sqrt(rootwt)` in the presence/absence of bloodworms. To minimize selection bias the `Variety` main effects and `Tmt:Variety` interactions are taken as random. The main effect of treatment is fitted as fixed to allow for the likely scenario that rather than a single population of treatment by variety effects there are in fact two populations (control and treated) with a different mean for each. There is evidence of this prior to analysis with the large difference in mean `sqrt(rootwt)` for the two groups (14.93 and 8.23 for control and treated respectively). The inclusion of `Tmt` as a fixed effect ensures that E-BLUPs of `Tmt:Variety` effects are shrunk to the correct mean (treatment means rather than an overall mean).

The model for the data is given by

$$\boldsymbol{y} = \boldsymbol{X}\boldsymbol{\tau} + \boldsymbol{Z}_1\boldsymbol{u}_1 + \boldsymbol{Z}_2\boldsymbol{u}_2 + \boldsymbol{Z}_3\boldsymbol{u}_3 + \boldsymbol{Z}_4\boldsymbol{u}_4 + \boldsymbol{Z}_5\boldsymbol{u}_5 + \boldsymbol{e} \tag{7.8}$$

where $\boldsymbol{y}$ is a vector of length $n = 264$ containing the `sqrtroot` values, $\boldsymbol{\tau}$ corresponds to a constant term and the fixed treatment contrast and $\boldsymbol{u}_1 \ldots \boldsymbol{u}_5$ correspond to random `Variety`, `Tmt:Variety`, `Run`, `Tmt:Run` and `Variety:Run` effects. The random effects and error are assumed to be independent Gaussian variables with zero means and variance structures $\text{var}(\boldsymbol{u}_i) = \sigma_i^2 \boldsymbol{I}_{b_i}$ (where b_i is the length of $\boldsymbol{u}_i$, $i = 1 \ldots 5$) and $\text{var}(\boldsymbol{e}) = \sigma^2 \boldsymbol{I}_n$.

The ASReml-R call is:

```
asreml.options(gammaPar = TRUE)
rice1.asr <- asreml(sqrtroot ~ Tmt, random = ~idv(Variety) + idv(Variety):id(Tmt) +
    idv(Run) + idv(Pair) + idv(Run):id(Tmt), residual = ~idv(units), data = rice)
```

```
summary(rice1.asr)$loglik
```

```
[1] -345.2559
```

```
summary(rice1.asr)$varcomp
```

```
                    component std.error   z.ratio bound %ch
Variety!Variety     2.3778037 0.7911195 3.0056189     P 0.2
Run!Run             0.3217226 0.5483794 0.5866789     P 0.9
Variety:Tmt!Variety 0.4923122 0.2764192 1.7810344     P 0.0
Pair!Pair           0.9758347 0.3882604 2.5133509     P 0.1
Run:Tmt!Run         1.7478110 0.4793497 3.6462126     P 0.0
units!units         1.3149769        NA        NA     F 0.0
units!R             1.3149769 0.2974428 4.4209405     P 0.0
```

```
wald(rice1.asr)
```

```
Wald tests for fixed effects.
Response: sqrtroot
Terms added sequentially; adjusted for those above.

              Df Sum of Sq Wald statistic Pr(Chisq)
(Intercept)    1   1953.60        1485.65 < 2.2e-16 ***
Tmt            1    617.16         469.33 < 2.2e-16 ***
residual (MS)         1.31
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The estimated variance components from this analysis also appear in column (a) of Table 7.8. The variance component for the `Variety` main effects is large. There is evidence of `Variety:Tmt` interactions so we may expect some discrimination between varieties in terms of tolerance to bloodworms.

Given the large difference ($p < 0.001$) between `Tmt` means we may wish to allow for heterogeneity of variance associated with `Tmt`. Thus we fit a separate `Variety:Tmt` variance for each level of `Tmt` so that instead of assuming $\text{var}(\boldsymbol{u}_2) = \sigma_2^2 \boldsymbol{I}_{88}$ we assume

$$\text{var}(\boldsymbol{u}_2) = \begin{bmatrix} \sigma_{2c}^2 & 0 \\ 0 & \sigma_{2t}^2 \end{bmatrix} \otimes \boldsymbol{I}_{44}$$

where σ_{2c}^2 and σ_{2t}^2 are the `Variety:Tmt` interaction variances for control and treated respectively. This model can be fitted using a diagonal variance structure for the treatment part of the interaction. We also fit a separate `Run:Tmt` variance for each level of `Tmt` and heterogeneity at the residual level, by including an extra `at(Tmt,2):units` term. We have chosen level 2 of `Tmt` as we expect more variation for the exposed treatment and thus the extra variance component for this term should be positive.

By default, ASReml-R sets the parameter constraint for variance components to **P**ositive. To allow for negative components, which may have meaning in this particular example, we must set the parameter constraints to **U**nconstrained. The following sequence of calls

- creates default $\mathbf{R}$ and $\mathbf{G}$ parameter list objects (`start.values=T`) in `temp`
- opens the default text editor where the parameter constraints can be changed to **U** and the result saved to `RG.rice`
- fits the model using the $\mathbf{G}$ level parameter settings in `RG.rice` through the `G.param` argument.

```
asreml.options(gammaPar = TRUE)
temp.gam <- asreml(sqrtroot ~ Tmt, random = ~idv(Variety) + id(Variety):diag(Tmt) +
    idv(Run) + idv(Pair) + id(Run):diag(Tmt) + at(Tmt, 2):units, residual = ~idv(units), data
    = rice, start.values = TRUE)$vparameters.table
temp.gam
```

```
                Component Value Constraint
1          Variety!Variety   0.1          P
```

```
2  Variety:Tmt!Tmt_Control   0.1          P
3  Variety:Tmt!Tmt_Exposed   0.1          P
4                  Run!Run   0.1          P
5                Pair!Pair   0.1          P
6      Run:Tmt!Tmt_Control   0.1          P
7      Run:Tmt!Tmt_Exposed   0.1          P
8   at(Tmt, Exposed):units   0.1          P
9                  units!R   1.0          P
10             units!units   1.0          F
```

```
temp.gam[c(2, 3), "Constraint"] <- c("U", "U")
temp.gam[c(6, 7), "Constraint"] <- c("U", "U")
temp.gam
```

```
                Component Value Constraint
1          Variety!Variety   0.1          P
2  Variety:Tmt!Tmt_Control   0.1          U
3  Variety:Tmt!Tmt_Exposed   0.1          U
4                  Run!Run   0.1          P
5                Pair!Pair   0.1          P
6      Run:Tmt!Tmt_Control   0.1          U
7      Run:Tmt!Tmt_Exposed   0.1          U
8   at(Tmt, Exposed):units   0.1          P
9                  units!R   1.0          P
10             units!units   1.0          F
```

```
asreml.options(gammaPar = TRUE)
rice2.asr <- asreml(sqrtroot ~ Tmt, random = ~idv(Variety) + id(Variety):diag(Tmt) +
    idv(Run) + idv(Pair) + id(Run):diag(Tmt) + at(Tmt, 2):idv(units), residual = ~idv(units),
    G.param = temp.gam, data = rice)
```

```
summary(rice2.asr)$loglik
```

```
[1] -343.2199
```

```
summary(rice2.asr)$varcomp
```

```
                              component std.error    z.ratio bound %ch
Variety!Variety               2.3341384 0.7761558  3.0073065     P 0.1
Run!Run                       0.3193663 0.5435029  0.5876075     P 0.5
Variety:Tmt!Tmt_Control       1.5055897 0.6644957  2.2657628     U 0.2
Variety:Tmt!Tmt_Exposed      -0.3721460 0.4563015 -0.8155702     U 0.5
Pair!Pair                     0.9875966 0.3811214  2.5912913     P 0.1
Run:Tmt!Tmt_Control           1.3891313 0.6359438  2.1843616     U 0.0
Run:Tmt!Tmt_Exposed           2.2242074 0.7237749  3.0730651     U 0.1
at(Tmt, Exposed):units!units  0.2039961 0.6317102  0.3229267     P 0.3
units!units                   1.1565111        NA         NA     F 0.0
units!R                       1.1565111 0.4173686  2.7709588     P 0.0
```

```
wald(rice2.asr)
```

```
Wald tests for fixed effects.
```

```
Response: sqrtroot
Terms added sequentially; adjusted for those above.

              Df Sum of Sq Wald statistic Pr(Chisq)
(Intercept)    1   1477.33        1277.40 < 2.2e-16 ***
Tmt            1    519.19         448.93 < 2.2e-16 ***
residual (MS)         1.16
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Table 7.8: Estimated variance components from univariate analyses of bloodworm data. (a) Model with homogeneous variance for all terms and (b) model with heterogeneous variance for interactions involving tmt

	(a)	(b)	
	homogeneous	heterogeneous	
source		control	treated
Variety	2.378	2.333	
Variety:Tmt	0.492	1.505	-0.372
Run	0.321	0.319	
Run:Tmt	1.748	1.389	2.224
Variety:Run (Pair)	0.976	0.987	
Tmt:Pair	1.315	1.156	1.360
REML log-likelihood	-345.256	-343.22	

The estimated variance components from this analysis are given in column (b) of Table 7.8. There is no significant variance heterogeneity at the residual or `Run:Tmt` level. This indicates that the square root transformation of the data has successfully stabilised the error variance. There is, however, significant variance heterogeneity for `Variety:Tmt` interactions with the variance being much greater for the control group. This reflects the fact that in the absence of bloodworms the potential maximum root area is greater. Note that the `Variety:Tmt` interaction variance for the treated group is negative. The negative component is meaningful (and in fact necessary and obtained by changing the constraint codes for variance parameters to `U` as described above) in this context since it should be considered as part of the variance structure for the combined variety main effects and treatment by variety interactions. That is,

$$\text{var}\left(\mathbf{1}_2 \otimes \boldsymbol{u}_1 + \boldsymbol{u}_2\right) = \begin{bmatrix} \sigma_1^2 + \sigma_{2c}^2 & \sigma_1^2 \\ \sigma_1^2 & \sigma_1^2 + \sigma_{2t}^2 \end{bmatrix} \otimes \boldsymbol{I}_{44} \tag{7.9}$$

Using the estimates from Table 7.8 this structure is estimated as

$$\begin{bmatrix} 3.84 & 2.33 \\ 2.33 & 1.96 \end{bmatrix} \otimes \boldsymbol{I}_{44}$$

Thus the variance of the variety effects in the control group (also known as the genetic variance for this group) is 3.84. The genetic variance for the treated group is much lower (1.96). The

genetic correlation is $2.33/\sqrt{3.84 \times 1.96} = 0.85$ which is strong, supporting earlier indications of the dependence between the treated and control root area (Figure 7.8).

A multivariate approach

In this simple case in which the variance heterogeneity is associated with the two level factor `Tmt`, the analysis is equivalent to a bivariate analysis in which the two traits correspond to the two levels of `Tmt`, namely `sqrtroot` for control and treated. The model for each trait is given by

$$\boldsymbol{y}_j = \boldsymbol{X}\boldsymbol{\tau}_j + \boldsymbol{Z}_v\boldsymbol{u}_{v_j} + \boldsymbol{Z}_r\boldsymbol{u}_{r_j} + \boldsymbol{e}_j \quad (j = c, t) \tag{7.10}$$

where $\boldsymbol{y}_j$ is a vector of length $n = 132$ containing the `sqrtroot` values for variate j ($j = c$ for control and $j = t$ for treated), $\boldsymbol{\tau}_j$ corresponds to a constant term and $\boldsymbol{u}_{v_j}$ and $\boldsymbol{u}_{r_j}$ correspond to random variety and run effects. The design matrices are the same for both traits. The random effects and error are assumed to be independent Gaussian variables with zero means and variance structures $\text{var}\left(\boldsymbol{u}_{v_j}\right) = \sigma^2_{v_j}\boldsymbol{I}_{44}$, $\text{var}\left(\boldsymbol{u}_{r_j}\right) = \sigma^2_{r_j}\boldsymbol{I}_{66}$ and $\text{var}\left(\boldsymbol{e}_j\right) = \sigma^2_j\boldsymbol{I}_{132}$. The bivariate model can be written as a direct extension of (7.10), namely

$$\boldsymbol{y} = \left(\boldsymbol{I}_2 \otimes \boldsymbol{X}\right)\boldsymbol{\tau} + \left(\boldsymbol{I}_2 \otimes \boldsymbol{Z}_v\right)\boldsymbol{u}_v + \left(\boldsymbol{I}_2 \otimes \boldsymbol{Z}_r\right)\boldsymbol{u}_r + \boldsymbol{e}^* \tag{7.11}$$

where $\boldsymbol{y} = (\boldsymbol{y}_c', \boldsymbol{y}_t')'$, $\boldsymbol{u}_v = \left(\boldsymbol{u}_{v_c}', \boldsymbol{u}_{v_t}'\right)'$, $\boldsymbol{u}_r = \left(\boldsymbol{u}_{r_c}', \boldsymbol{u}_{r_t}'\right)'$ and $\boldsymbol{e}^* = (\boldsymbol{e}_c', \boldsymbol{e}_t')'$.

There is an equivalence between the effects in this bivariate model and the univariate model of (7.8). The variety effects for each trait ($\boldsymbol{u}_v$ in the bivariate model) are partitioned in (7.8) into variety main effects and `tmt.variety` interactions so that $\boldsymbol{u}_v = \boldsymbol{1}_2 \otimes \boldsymbol{u}_1 + \boldsymbol{u}_2$. There is a similar partitioning for the run effects and the errors (Table 7.9).

Table 7.9: Equivalence of random effects in bivariate and univariate analyses

effects	bivariate (model 7.11)	univariate (model 7.8)
trait:Variety	$\boldsymbol{u}_v$	$\boldsymbol{1}_2 \otimes \boldsymbol{u}_1 + \boldsymbol{u}_2$
trait:Run	$\boldsymbol{u}_r$	$\boldsymbol{1}_2 \otimes \boldsymbol{u}_3 + \boldsymbol{u}_4$
Pair:trait	$\boldsymbol{e}^*$	$\boldsymbol{1}_2 \otimes \boldsymbol{u}_5 + \boldsymbol{e}$

In addition to the assumptions in the models for individual traits (7.10), the bivariate analysis involves the assumptions $cov\left(\boldsymbol{u}_{v_c}\right)\boldsymbol{u}_{v_t}' = \sigma_{v_{ct}}\boldsymbol{I}_{44}$, $cov\left(\boldsymbol{u}_{r_c}\right)\boldsymbol{u}_{r_t}' = \sigma_{r_{ct}}\boldsymbol{I}_{66}$ and $cov\left(\boldsymbol{e}_c\right)\boldsymbol{e}_t' = \sigma_{ct}\boldsymbol{I}_{132}$. Thus random effects and errors are correlated between traits. So, for example, the variance matrix for the variety effects for each trait is given by

$$\text{var}\left(\boldsymbol{u}_v\right) = \begin{bmatrix} \sigma^2_{v_c} & \sigma_{v_{ct}} \\ \sigma_{v_{ct}} & \sigma^2_{v_t} \end{bmatrix} \otimes \boldsymbol{I}_{44}$$

This unstructured form for `trait:Variety` in the bivariate analysis is equivalent to the `Variety` main effect plus heterogeneous `Variety:Tmt` interaction variance structure (7.9) in the univariate

analysis. Similarly the unstructured form for `trait:Run` is equivalent to the `Run` main effect plus heterogeneous `Run:Tmt` interaction variance structure. The unstructured form for the errors (`Pair:trait`) in the bivariate analysis is equivalent to the `Pair` plus heterogeneous error (`Pair:Tmt`) variance in the univariate analysis.

The ASReml-R call is:

```
riceMV.asr <- asreml(cbind(syc, sye) ~ trait, random = ~us(trait):id(Variety) +
    us(trait):id(Run), residual = ~id(units):us(trait), data = riceMV)
```

```
Warning in asreml(cbind(syc, sye) ~ trait, random = ~us(trait):id(Variety) + :  Some components
changed by more than 1% on the last iteration.
```

```
summary(riceMV.asr)$loglik
```

```
[1] -343.22
```

```
summary(riceMV.asr)$varcomp
```

```
                           component std.error   z.ratio bound %ch
trait:Variety!trait_syc:syc 3.8370236 1.1041043 3.4752367     P 0.3
trait:Variety!trait_sye:syc 2.3317436 0.7741058 3.0121769     P 0.3
trait:Variety!trait_sye:sye 1.9598665 0.7268965 2.6962112     P 0.3
trait:Run!trait_syc:syc     1.7090708 0.6541103 2.6128176     P 0.3
trait:Run!trait_sye:syc     0.3207441 0.5443446 0.5892299     P 1.7
trait:Run!trait_sye:sye     2.5450260 0.7965487 3.1950665     P 0.2
units:trait!R               1.0000000        NA        NA     F 0.0
units:trait!trait_syc:syc   2.1435663 0.4820833 4.4464646     P 0.0
units:trait!trait_sye:syc   0.9870784 0.3809549 2.5910636     P 0.1
units:trait!trait_sye:sye   2.3471096 0.5073875 4.6258716     P 0.0
```

```
wald(rice2.asr)
```

```
Wald tests for fixed effects.
Response: sqrtroot
Terms added sequentially; adjusted for those above.

             Df Sum of Sq Wald statistic Pr(Chisq)
(Intercept)   1   1477.33        1277.40 < 2.2e-16 ***
Tmt           1    519.19         448.93 < 2.2e-16 ***
residual (MS)       1.16
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The resultant REML log-likelihood is identical to that of the heterogeneous univariate analysis (column (b) of Table 7.8). The estimated variance parameters are summarised in Table 7.10.

Predicted variety means are obtained from:

```
riceMV.pv <- predict(riceMV.asr, classify = "trait:Variety")
```

Table 7.10: Estimated variance components from bivariate analysis of bloodworm data

source	control variance	treated variance	covariance
us(trait):Variety	3.84	1.96	2.33
us(trait):Run	1.71	2.54	0.32
Pair:us(trait)	2.14	2.35	0.99

```
head(riceMV.pv$pvals, 20)
```

```
Notes:
- The predictions are obtained by averaging across the hypertable
  calculated from model terms constructed solely from factors in
  the averaging and classify sets.
- Use 'average' to move ignored factors into the averaging set.
- The ignored set: Run
```

```
         Variety trait predicted.value std.error    status
1       AliCombo   syc       14.953221 0.9180989 Estimable
2       AliCombo   sye        7.994086 0.7993077 Estimable
3         Amaroo   syc       16.161246 0.9181010 Estimable
4         Amaroo   sye        8.481302 0.7992850 Estimable
5        Balilla   syc       14.420180 0.9185728 Estimable
6        Balilla   sye        8.230838 0.7995129 Estimable
7      Bluebelle   syc       13.103200 0.9309795 Estimable
8      Bluebelle   sye        6.629743 0.8062304 Estimable
9          Bogan   syc       15.768252 0.9548615 Estimable
10         Bogan   sye        8.007108 0.8190104 Estimable
11           C22   syc       16.667976 0.9180998 Estimable
12           C22   sye        8.954416 0.7993082 Estimable
13       Calrose   syc       15.865773 0.9179556 Estimable
14       Calrose   sye        9.506738 0.7992441 Estimable
15 Cent.Patna231   syc       12.771980 0.9299189 Estimable
16 Cent.Patna231   sye        6.767878 0.8056707 Estimable
17   Chiyohikari   syc       17.339381 0.9300733 Estimable
18   Chiyohikari   sye        9.643430 0.8057379 Estimable
19          Dawn   syc       12.209055 0.9841713 Estimable
20          Dawn   sye        6.540125 0.8412630 Estimable
```

These predictions are on the square root scale; it is straightforward to *back-transform* the predicted means to the original scale of measurement. Approximate standard errors on the original scale can be calculated from a Taylor series approximation. That is, if x is a random variable with $E(x) = \theta$, and $y = g(x)$ is some function of x, then $var(y) = (dy/dx)^2_\theta var(x)$. See Kendall and Stuart (1969) pp 231, for example. In this case, $g(x) = x^2$ and $g'(x) = dy/dx = 2x$. The following code calculates the transformed predictions and approximate standard errors:

```
pv <- riceMV.pv$pvals
pv$rootwt <- pv$predicted.value^2
```

```
pv$approxSE <- sqrt(4 * pv$predicted.value^2 * pv$std.error^2)
pv$est.status <- NULL
head(pv, 20)
```

```
Notes:
- The predictions are obtained by averaging across the hypertable
  calculated from model terms constructed solely from factors in
  the averaging and classify sets.
- Use 'average' to move ignored factors into the averaging set.
- The ignored set: Run
```

	Variety	trait	predicted.value	std.error	status	rootwt	approxSE
1	AliCombo	syc	14.953221	0.9180989	Estimable	223.59883	27.45707
2	AliCombo	sye	7.994086	0.7993077	Estimable	63.90540	12.77947
3	Amaroo	syc	16.161246	0.9181010	Estimable	261.18589	29.67531
4	Amaroo	sye	8.481302	0.7992850	Estimable	71.93248	13.55796
5	Balilla	syc	14.420180	0.9185728	Estimable	207.94158	26.49197
6	Balilla	sye	8.230838	0.7995129	Estimable	67.74670	13.16132
7	Bluebelle	syc	13.103200	0.9309795	Estimable	171.69384	24.39762
8	Bluebelle	sye	6.629743	0.8062304	Estimable	43.95349	10.69020
9	Bogan	syc	15.768252	0.9548615	Estimable	248.63777	30.11299
10	Bogan	sye	8.007108	0.8190104	Estimable	64.11378	13.11581
11	C22	syc	16.667976	0.9180998	Estimable	277.82144	30.60573
12	C22	sye	8.954416	0.7993082	Estimable	80.18157	14.31468
13	Calrose	syc	15.865773	0.9179556	Estimable	251.72274	29.12815
14	Calrose	sye	9.506738	0.7992441	Estimable	90.37807	15.19641
15	Cent.Patna231	syc	12.771980	0.9299189	Estimable	163.12346	23.75381
16	Cent.Patna231	sye	6.767878	0.8056707	Estimable	45.80418	10.90536
17	Chiyohikari	syc	17.339381	0.9300733	Estimable	300.65412	32.25379
18	Chiyohikari	sye	9.643430	0.8057379	Estimable	92.99574	15.54015
19	Dawn	syc	12.209055	0.9841713	Estimable	149.06102	24.03160
20	Dawn	sye	6.540125	0.8412630	Estimable	42.77323	11.00393

Interpretation of results

Recall that the primary interest is varietal tolerance to bloodworms. This could be defined in various ways: One option is to consider the regression implicit in the variance structure for the trait by variety effects. The variance structure can arise from a regression of treated variety effects on control effects, namely

$$\boldsymbol{u}_{v_t} = \beta \boldsymbol{u}_{v_c} + \boldsymbol{\epsilon}$$

where the slope $\beta = \sigma_{v_{ct}}/\sigma^2_{v_c}$.

Tolerance can be defined in terms of the deviations from regression, $\boldsymbol{\epsilon}$. Varieties with large positive deviations have greatest tolerance to bloodworms. Note that this is similar to the original approach except that the regression has been conducted at the genotypic rather than the phenotypic level. In Figure 7.9 the E-BLUPs for treated have been plotted against the E-BLUPs for control for each variety and the fitted regression line (slope = 0.61) has been drawn. Varieties with large positive deviations from the regression line include YRK3, Calrose, HR19 and WC1403.

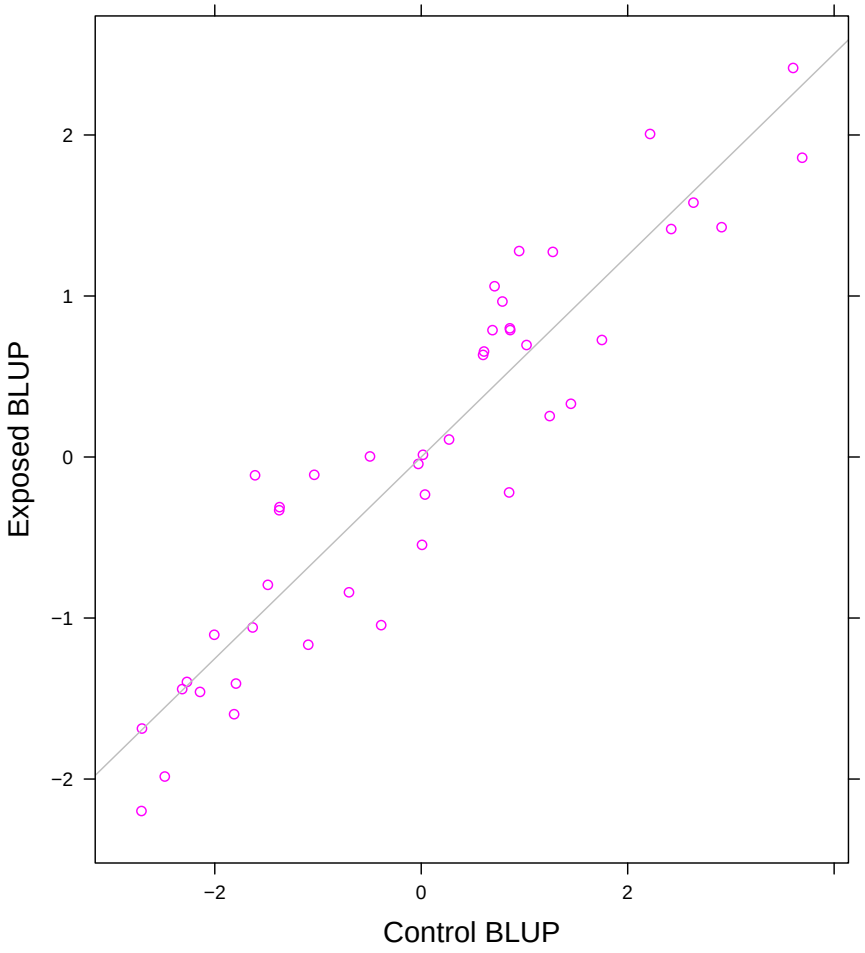

Figure 7.9: E-BLUPs for treated plotted against E-BLUPs for control

An alternative definition of tolerance is the simple difference between treated and control E-BLUPs for each variety, namely $\boldsymbol{\delta} = \boldsymbol{u}_{v_c} - \boldsymbol{u}_{v_t}$. Unless $\beta = 1$ the two measures $\boldsymbol{\epsilon}$ and $\boldsymbol{\delta}$ have very different interpretations. The key difference is that $\boldsymbol{\epsilon}$ is a measure which is *independent* of inherent vigour whereas $\boldsymbol{\delta}$ is not. To see this consider

$$\begin{aligned} cov\left(\boldsymbol{\epsilon}\right)\boldsymbol{u}'_{v_c} &= cov\left(\boldsymbol{u}_{v_t} - \beta\boldsymbol{u}_{v_c}\right)\boldsymbol{u}'_{v_c} \\ &= \left(\sigma_{v_{ct}} - \frac{\sigma_{v_{ct}}}{\sigma^2_{v_c}}\sigma^2_{v_c}\right)\boldsymbol{I}_{44} \\ &= \mathbf{0} \end{aligned}$$

whereas

$$\begin{aligned} cov\left(\boldsymbol{\delta}\right)\boldsymbol{u}'_{v_c} &= cov\left(\boldsymbol{u}_{v_c} - \boldsymbol{u}_{v_t}\right)\boldsymbol{u}'_{v_c} \\ &= \left(\sigma^2_{v_c} - \sigma_{v_{ct}}\right)\boldsymbol{I}_{44} \end{aligned}$$

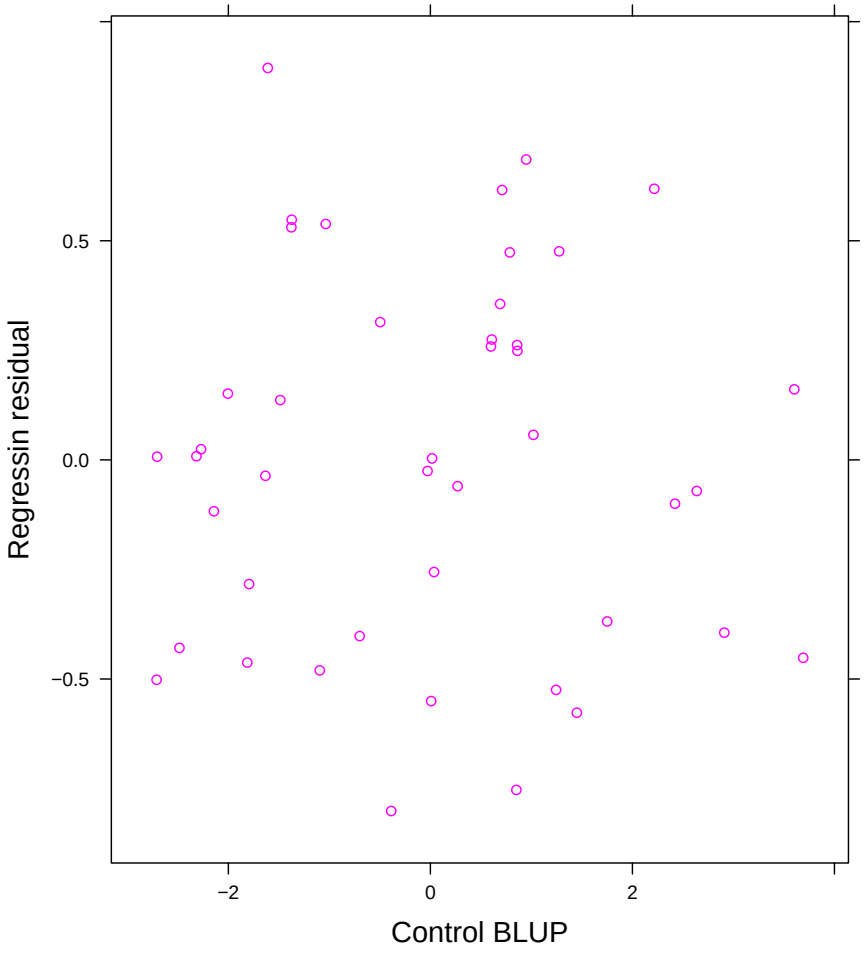

Figure 7.10: Estimated deviations from regression of treated on control for each variety plotted against estimate for control

The independence of $\boldsymbol{\epsilon}$ and $\boldsymbol{u}_{v_c}$ and dependence between $\boldsymbol{\delta}$ and $\boldsymbol{u}_{v_c}$ is clearly illustrated in Figures 7.10 and 7.11. In this example the two measures have provided very different rankings of the varieties. The choice of tolerance measure depends on the aim of the experiment. In this experiment the aim was to identify tolerance which is independent of inherent vigour so the deviations from regression is preferred.

7.9 Balanced longitudinal data - random coefficients and cubic smoothing splines

This section illustrates the use of random coefficients and cubic smoothing splines for the analysis of balanced longitudinal data.

The implementation of cubic smoothing splines in ASReml-R is based on the mixed model formulation of Verbyla et al. (1999). More recently the methodology has been extended so that the user can specify knot points; in the original approach the knot points were taken to be the ordered set

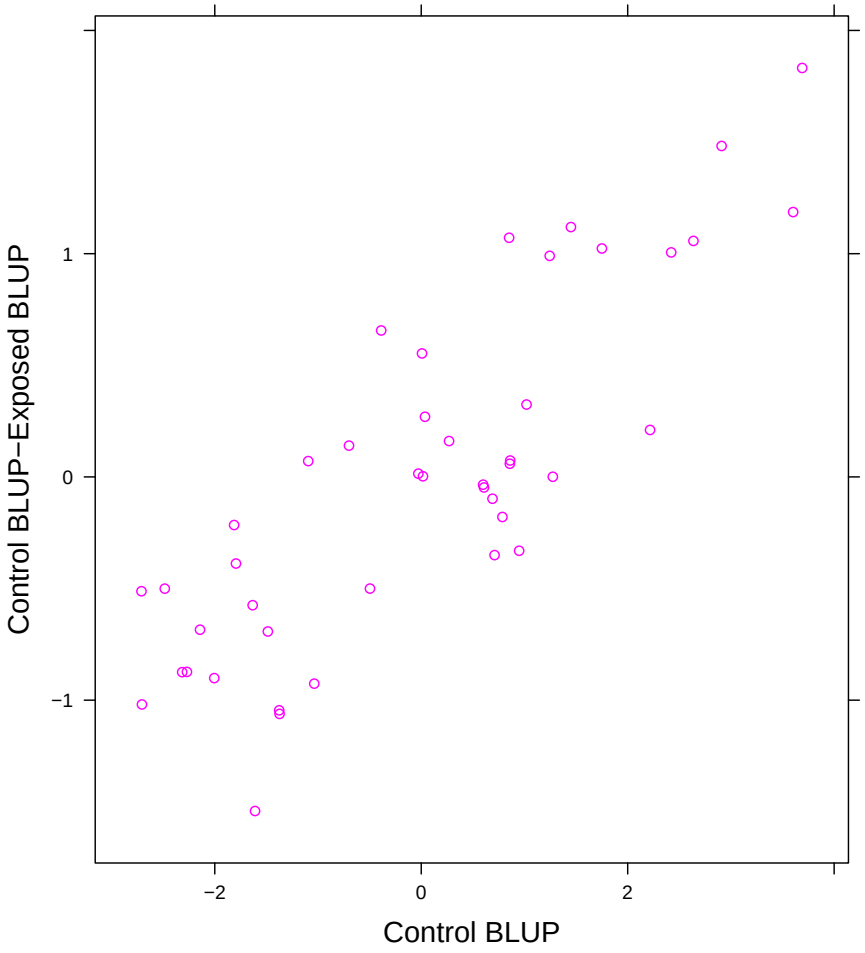

Figure 7.11: Estimated difference between control and treated for each variety plotted against estimate for control

of unique values of the explanatory variable. The specification of knot points is particularly useful if the number of unique values in the explanatory variable is large, or if units are measured at different times.

These data were originally reported by Draper and Smith (1998, ex24N, p559) and have recently been re-analysed by Pinheiro and Bates (2000, p338). The data are trunk circumferences (in millimetres) of each of 5 trees taken at 7 times (Figure 7.12). All trees were measured at the same time so that the data are balanced. The aim of the study is unclear, though both previous analyses involved modelling the overall *growth* curve, accounting for the obvious variation in both level and shape between trees.

Pinheiro and Bates (2000) used a nonlinear mixed effects modelling approach, in which they modelled the growth curves by a three parameter logistic function of age:

$$y = \frac{\phi_1}{1 + \exp\left[-(x - \phi_2)/\phi_3\right]} \tag{7.12}$$

where y is the trunk circumference, x is the tree age in days since December 31 1968, ϕ_1 is the asymptotic height, ϕ_2 is the inflection point or the time at which the tree reaches $0.5\phi_1$, ϕ_3 is the time elapsed between trees reaching half and about 3/4 of ϕ_1.

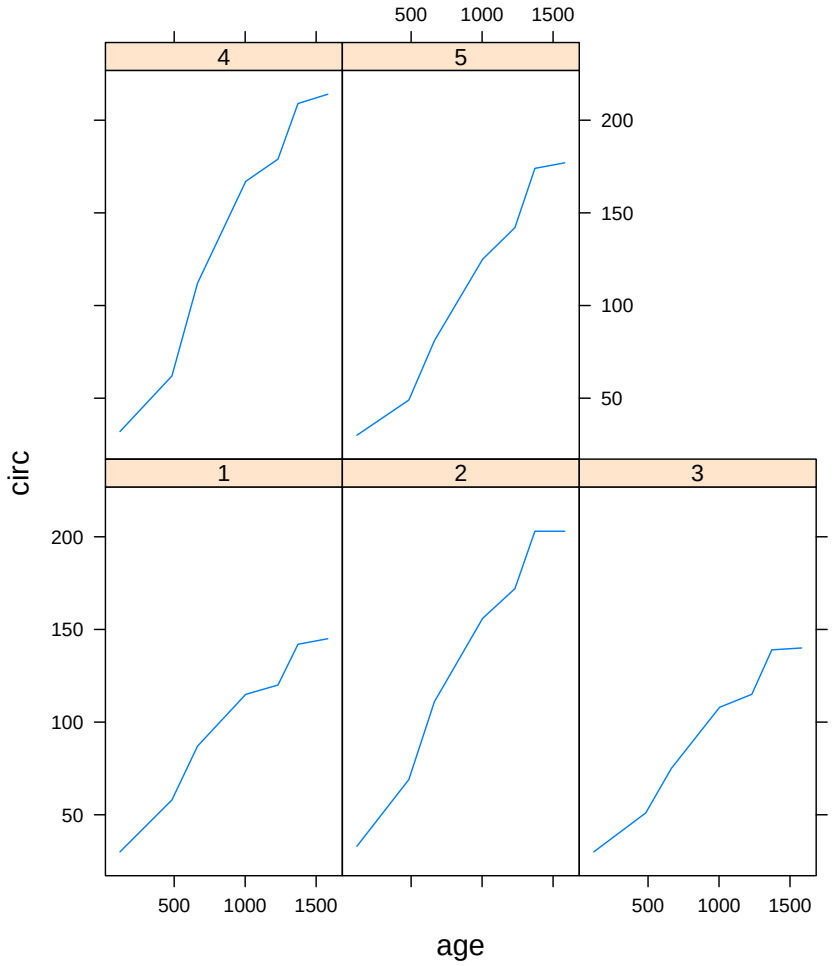

Figure 7.12: Trellis plot of trunk circumference (mm) for each tree against age in days since 1 December 1968.

The data frame `orange` contains:

```
orange <- asreml.read.table("../examples/orange.csv", header = T, sep = ",")
```

```
names(orange)
```

```
[1] "Tree"   "x"      "circ"   "Season"
```

where `Tree` is a factor with 5 levels, `x` is tree age in days since 31 December 1968, `circ` is the trunk circumference and `Season` is a factor with two levels, `Spring` and `Autumn`. The factor `Season` was included after noting that tree age spans several years and if converted to day of year, measurements were taken in either April/May (`Spring`) or September/October (`Autumn`).

7.9 Balanced longitudinal data

Initially we restrict the dataset to tree 1 to demonstrate fitting cubic splines in ASReml-R. The model includes the intercept and linear regression of trunk circumference on x and an additional random term `spl(x)` which includes a random term with a special design matrix with $7 - 2 = 5$ columns which relate to the vector, $\boldsymbol{\delta}$ whose elements $\delta_i, i = 2, \ldots, 6$ are the second differentials of the cubic spline at the knot points. The second differentials of a natural cubic spline are zero at the first and last knot points (Green and Silverman; 1994).

```
asreml.options(gammaPar = TRUE)
orange.asr <- asreml(circ ~ x, random = ~spl(x), residual = ~idv(units), knot.points =
    list(x = c(118, 484, 664, 1004, 1231, 1372, 1582)), data = orange, subset = Tree == 1)
```

```
Warning in asreml(circ ~ x, random = ~spl(x), residual = ~idv(units), knot.points = list(x
= c(118, :  Some components changed by more than 1% on the last iteration.
```

```
orange.asr$trace
```

```
                    1            2            3
LogLik      -20.90425 -20.90127049 -20.90127049
Sigma2       48.46981  49.15204735  49.15204735
DF            5.00000   5.00000000   5.00000000
spl(x)        0.10000   0.09102007   0.07535652
units!units   1.00000   1.00000000   1.00000000
units!R       1.00000   1.00000000   1.00000000
```

In this example the spline knot points are specifically given in the `knot.points` argument. These extra points have no effect in this case as they are the seven ages existing in the data file. In this instance the analysis would be the same if the `knot.points` argument was omitted.

```
summary(orange.asr)$varcomp
```

```
             component std.error   z.ratio bound  %ch
spl(x)        3.703927  10.93709 0.3386573     P 17.2
units!units 49.152047        NA        NA     F  0.0
units!R     49.152047  36.83725 1.3343028     P  0.0
```

```
wald(orange.asr, denDF = "default")
```

```
Model fitted using the gamma parameterization.
ASReml 4.1.0 Mon Aug 12 10:35:07 2019
          LogLik        Sigma2     DF     wall    cpu
 1      -20.8998       50.5242      5 10:35:07    0.0
 2      -20.8997       50.4031      5 10:35:07    0.0
 3      -20.8996       50.1660      5 10:35:07    0.0

$Wald

Wald tests for fixed effects.
Response: circ

            Df denDF  F.inc         Pr
(Intercept)  1   3.5 1383.0 0.00001093
```

```
x            1   3.5  217.7 0.00027591

$stratumVariances
              df Variance   spl(x) units!R
spl(x)  1.487218 97.20763 11.91578       1
units!R 3.512782 50.16598  0.00000       1
```

Predicted values of the spline curve at nominated points can be obtained by:

```
orange.pv <- predict(orange.asr, classify = "x", design.points = list(x = seq(150, 1500,
    50)))
```

The `design.points` argument adds the nominated points to the design matrix for prediction purposes (Figure 7.13). Note that `design.points` could have been included in the call to asreml.options() instead of in `predict()`. If omitted from either `predict()` or asreml.options() a default set of points for prediction purposes would have been generated. The REML estimate of the smoothing constant and the fitted cubic smoothing spline (Figure 7.13) indicate there is some nonlinearity. The four points below the line were the spring measurements.

An analysis of variance decomposition for the full dataset is given in Table 7.11, following Verbyla et al. (1999).

Table 7.11: ANOVA decomposition for the orange data

stratum	decomposition	type	df or ne
(Intercept)	1	f	1
Age			
	x	f	1
	spl(x)	r	5
	residual	r	7
Tree			
	Tree	rc	5
Age:Tree			
	x:Tree	rc	5
	spl(x):Tree	r	25
error		r	

An overall spline is included as well as tree deviation splines. We note that the intercept and slope for the tree deviation splines are assumed to be random effects. This is consistent with Verbyla et al. (1999). In this sense the tree deviation splines play a role in modelling the conditional curves for each tree and variance modelling. The intercept and slope for each tree are included as random coefficients (denoted by **rc** in Table 7.11). Thus, if $\boldsymbol{U}^{5\times 2}$ is the matrix of intercepts (column 1) and slopes (column 2) for each tree, then we assume that

$$\text{var}\left(\text{vec}(\boldsymbol{U})\right) = \boldsymbol{\Sigma} \otimes \boldsymbol{I}_5$$

where $\boldsymbol{\Sigma}$ is a 2×2 symmetric positive definite matrix. Non smooth variation can be modelled at the overall mean (across trees) level and this is achieved by including the factor `dev(x)` as a random term. The full model is:

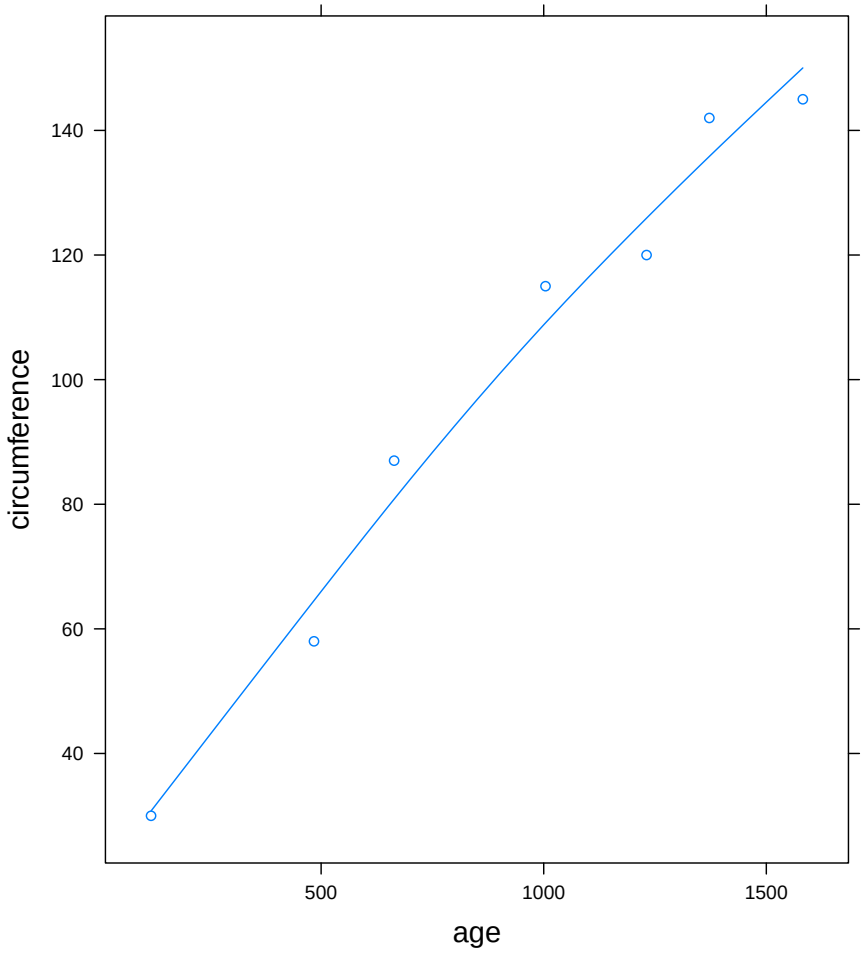

Figure 7.13: Fitted cubic smoothing spline for tree 1

```
asreml.options(gammaPar = TRUE)
orange1.asr <- asreml(circ ~ x, random = ~str(~Tree + x:Tree, ~diag(2):id(5)) + spl(x) +
    spl(x):id(Tree) + idv(dev(x)), residual = ~idv(units), knot.points = list(x = c(118, 484,
    664, 1004, 1231, 1372, 1582)), data = orange)
```

Table 7.12 presents the sequence of fitted models. We stress the importance of model building in these settings, where we generally commence with relatively simple variance models and update to more complex variance models if appropriate. Note that the REML log-likelihoods for models 1 and 2 are comparable and likewise for models 3 to 6. The REML log-likelihoods are not comparable between these groups because of the inclusion of the fixed `Season` factor.

We begin by modelling the variance matrix for the intercept and slope for each tree, $\mathbf{\Sigma}$, as a diagonal matrix as there is no point including a covariance component between the intercept and slope if the variance component(s) for one (or both) is zero. Model 1 also does not include a non-smooth component at the overall level (that is, `dev(x)`).

The ASReml-R call and variance components for model 1 are:

Table 7.12: Sequence of models fitted to the `orange` data

	model					
term	1	2	3	4	5	6
`Tree`	y	y	y	y	y	y
`x:Tree`	y	y	y	y	y	y
`cov(Tree, x:Tree)`	n	n	n	n	n	y
`spl(x)`	y	y	y	y	n	y
`spl(x):Tree`	y	y	y	n	y	y
`dev(x)`	n	y	y	n	n	n
`Season`	n	n	y	y	y	y
REML log-likelihood	-97.78	-94.07	-87.95	-91.22	-90.18	-87.43

```
orange1.asr <- asreml(circ ~ x, random = ~str(~Tree + x:Tree, ~diag(2):id(5)) + spl(x) +
    spl(x):id(Tree), residual = ~id(units), knot.points = list(x = c(118, 484, 664, 1004,
    1231, 1372, 1582)), data = orange)
```

```
summary(orange1.asr)$varcomp
```

```
                        component    std.error  z.ratio bound %ch
Tree+x:Tree!diag(2)_1 3.048865e+01 2.457266e+01 1.240755     P 0.4
Tree+x:Tree!diag(2)_2 5.980326e-04 4.240080e-04 1.410428     P 0.3
spl(x)                6.398248e+02 4.130439e+02 1.549048     P 0.3
spl(x):Tree           7.101353e+00 4.915670e+00 1.444636     P 0.7
units!R               6.374557e+00 3.658904e+00 1.742204     P 0.0
```

The fitted curves from this model are shown in Figure 7.14. The fit is unacceptable because the spline has picked up too much curvature, suggesting there may be systematic non-smooth variation at the overall level. This can be formally examined by including the `dev(x)` term as a random effect.

Model 2 increased the REML log-likelihood by 3.70 ($P < 0.05$) with the `spl(x)` smoothing constant approaching the boundary. The `Season` factor provides a possible explanation. When included in Model 3, `Season` has a Wald statistic of 107.3 ($P < 0.01$) and `dev(x)` becomes bounded. The spring measurements are lower than the autumn measurements so growth is slower in winter. Models 4 and 5 successively examined each term, indicating that both smoothing constants are significant. Model 6 includes the covariance parameter between the intercept and slope for each tree; this ensures that the model will be translation invariant. This model requires care in the choice of starting values. The ASReml-R call, illustrating an alternative method for specifying initial values, and the fitted components for model 6 are:

```
orange6.asr <- asreml(circ ~ x + Season, random = ~str(~Tree + x:Tree, ~us(2, init = c(5,
    -0.01, 1e-04)):id(5)) + spl(x) + spl(x):id(Tree), residual = ~id(units), knot.points =
    list(x = c(118, 484, 664, 1004, 1231, 1372, 1582)), data = orange)
```

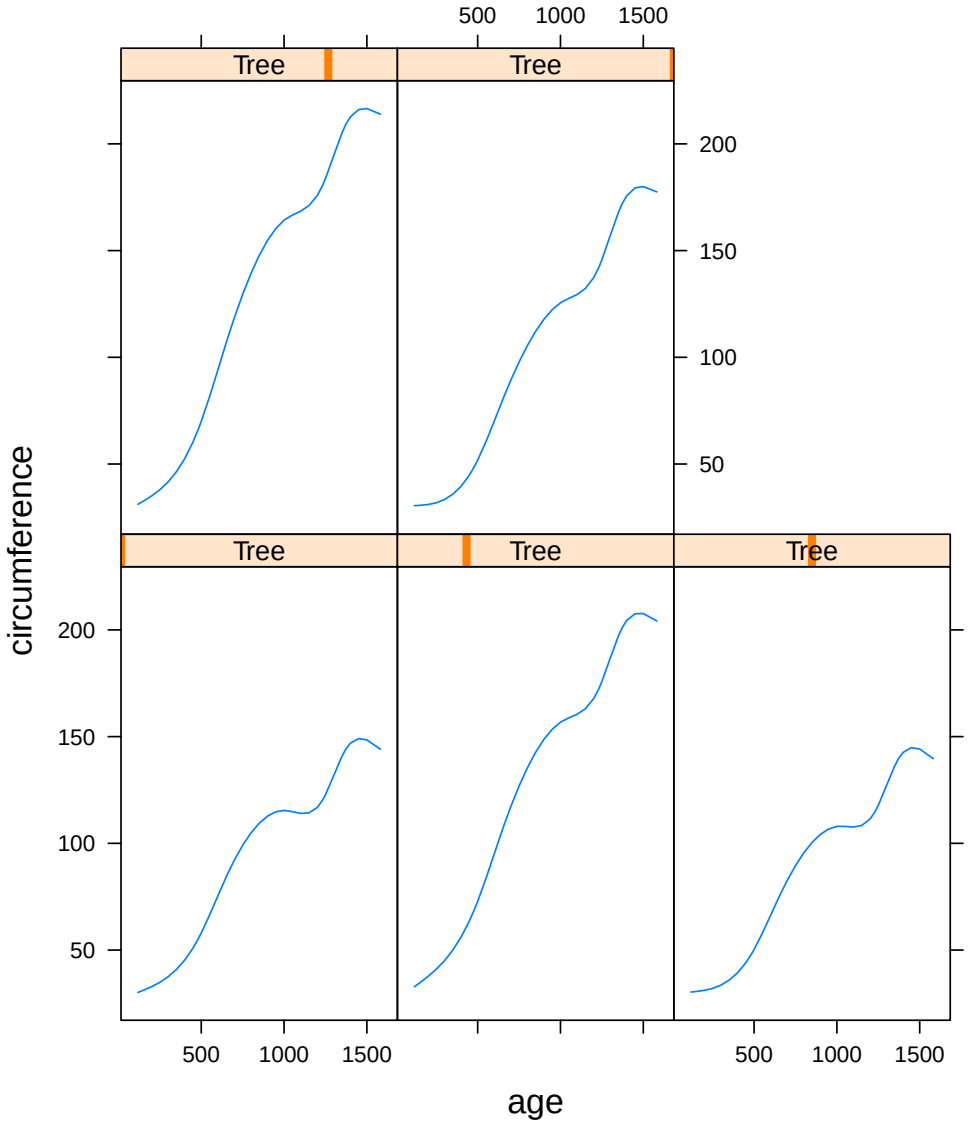

Figure 7.14: Plot of fitted cubic smoothing spline for model 1

```
summary(orange1.asr)$varcomp
```

```
                          component    std.error  z.ratio bound %ch
Tree+x:Tree!diag(2)_1 3.048865e+01 2.457266e+01 1.240755     P 0.4
Tree+x:Tree!diag(2)_2 5.980326e-04 4.240080e-04 1.410428     P 0.3
spl(x)                6.398248e+02 4.130439e+02 1.549048     P 0.3
spl(x):Tree           7.101353e+00 4.915670e+00 1.444636     P 0.7
units!R               6.374557e+00 3.658904e+00 1.742204     P 0.0
```

The fitted values for individual trees (adjusted for `Season`) from model 6 together with a marginal prediction and approximate confidence intervals (2× standard error of prediction) are shown in Figure 7.15. The conclusions from this analysis are quite different from those obtained by the nonlinear mixed effects analysis. The individual curves for each tree are not convincingly modelled

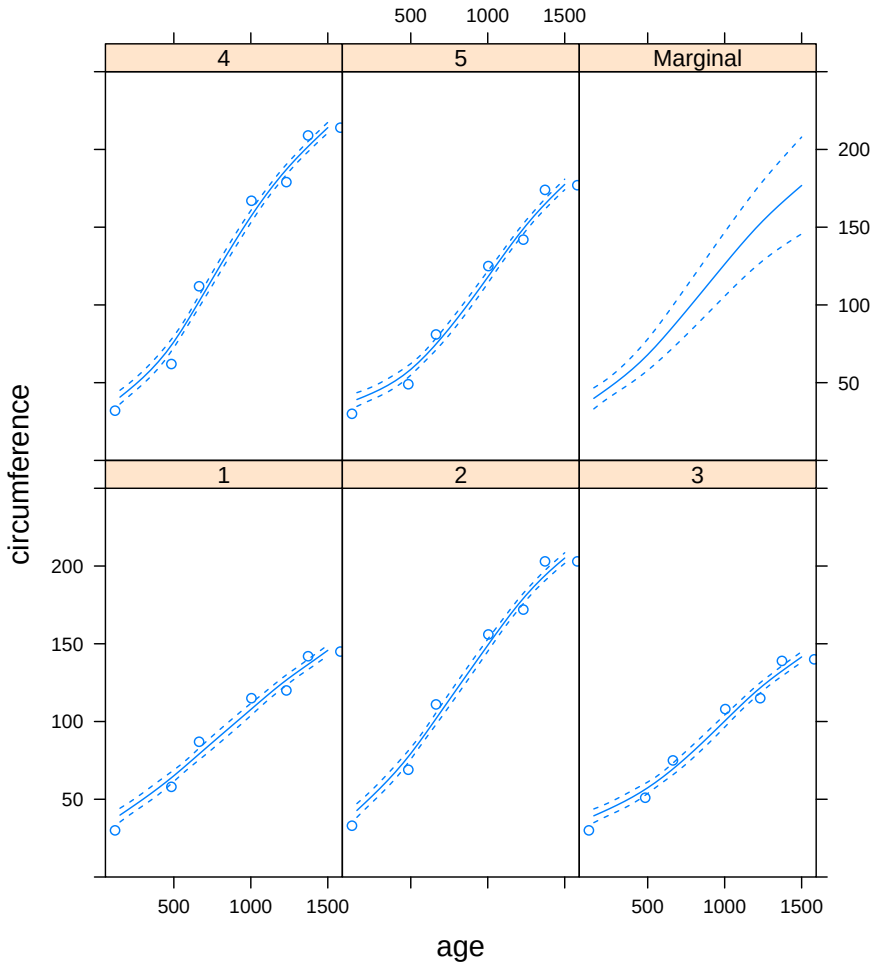

Figure 7.15: Fitted values adjusted for `Season` and approximate confidence intervals for model 6

by a logistic function. There is a distinct pattern in the residuals shown in Pinheiro and Bates (2000, p340), which is consistent for all trees; this is modelled here by the `Season` term.

A Some technical details about model fitting in ASReml-R

A.1 Sparse versus dense

The terms in the linear mixed model are partitioned into two sets; a dense set and a sparse set. The partition is defined by the fixed formula; all terms in the fixed formula are included in the dense set while terms in the random and sparse formulae are included in the sparse set. The inverse coefficient matrix is fully formed for the terms in the dense set which are fitted using dense equations. The inverse coefficient matrix is only partially formed for terms in the sparse set. Typically, the sparse set is large resulting in savings in memory and computing. A consequence is that the variance matrix of the BLUEs and BLUPs is only available for terms in the dense portion.

A.2 Ordering of terms in ASReml-R

Solutions for the fixed and random effects in linear mixed model analysis using ASReml-R are obtained by solving the corresponding mixed model equations in the numerical routines (Gilmour et al.; 1995). The sparse equations are processed first after being reordered to retain sparsity during solution. If keep.order=F, the remaining equations are processed with main effects before interactions and low order interactions before higher ones so that normal marginality of terms is achieved. The order of effects in the solution vector(s) in the returned object reflects the order of processing.

A.3 Aliasing and singularities

A singularity occurs when there is either

- a linear dependence in the model and therefore no information left to estimate the corresponding effect, or
- no data for that fixed effect,
- no data for a simple (uncorrelated) random effect.

The REML routines handle singularities by deleting the equations in question. Since the equations are solved from the bottom up, the first level (and hence the last level processed) of a factor is the one that will be declared singular and dropped from the model. The number of singularities is returned in the asreml object (`nsing`) and reported during the iterative process. Solutions that are

zero and have NA for their standard error are the singular effects.

Warning: Singularities in the *sparse* terms of the model may change with changes in the random terms included in the model. If this happens it will mean that changes in the REML log-likelihood are not valid for testing the changes made to the random model. A likelihood ratio test is not valid if the fixed model has changed.

A.3.1 Examples of aliasing

The sequence of examples in Table A.1 are presented to facilitate an understanding of over-parameterised models. It is assumed that Var is defined with 4 levels, Trt with 3 levels and Rep with 3 levels and that all Var:Trt combinations are present in the data.

Table A.1: Examples of aliasing

model	number of singularities	description
fixed = y ~ -1 + Var, random = ~ Rep	0	Var fully fitted
fixed = y ~ Var, random = ~ Rep	1	first level of Var dropped
fixed = y ~ -1 + Var + Trt, random = ~ Rep	1	Var fully specified, first level of Trt dropped from the models
fixed = y ~ Var + Trt + Var:Trt, random = ~ Rep	8	first level of both Var and Trt dropped from the model, together with subsequent interactions
fixed = ~ Var + Trt, random = ~ Rep, sparse = ~ Var:Trt	8	Var:Trt fully specified; (Intercept), Var and Trt completely singular and dropped from the model

B Available variance models

Table B.1 presents the full range of variance models available in ASReml-R with their algebraic descriptions and numbers of parameters. In most cases the algebraic form is for the correlation model (id() to agau()). However, the models from diag() onwards are additional heterogeneous variance models.

Recall from Section 4.2 the algebraic forms of the homogeneous and heterogeneous variance models are determined as follows. Let $\boldsymbol{C}^{(\omega\times\omega)} = [C_{ij}]$ be the correlation matrix for a particular correlation model. If $\boldsymbol{\Sigma}^{(\omega\times\omega)}$ is the corresponding homogeneous variance matrix then

$$\boldsymbol{\Sigma} \quad = \quad \sigma^2 \boldsymbol{C}$$

and has just one more parameter than the correlation model. For example, the homogeneous variance model corresponding to the id() correlation model has variance matrix $\boldsymbol{\Sigma} = \sigma^2 I_\omega$ (specified idv() in the ASReml-R function call, see below) and one parameter. Likewise, if $\boldsymbol{\Sigma}_h^{(\omega\times\omega)}$ is the heterogeneous variance matrix corresponding to $\boldsymbol{C}$, then

$$\boldsymbol{\Sigma}_h \quad = \quad \boldsymbol{DCD}$$

where $\boldsymbol{D}^{(\omega\times\omega)} = diag(\sigma_i)$. In this case there are an additional ω parameters. For example, the ASReml-R function for the heterogeneous variance model corresponding to id() variance model has variance matrix

$$\boldsymbol{\Sigma}_h \quad = \quad \begin{bmatrix} \sigma_1^2 & 0 & \dots & 0 \\ 0 & \sigma_2^2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \sigma_\omega^2 \end{bmatrix}$$

(specified idh() in the asreml() call, see below) and involves the ω parameters $\sigma_1^2 \dots \sigma_\omega^2$.

B Available variance models

Table B.1: Details of the available variance models

function name	description	algebraic form	number of parameters		
			corr	hom variance	het variance
Correlation models					
id()	identity	$C_{ii} = 1,\ C_{ij} = 0,\ \ i \neq j$	0	1	ω
ar1()	1^{st} order autoregressive	$C_{ii} = 1,\ C_{i+1,i} = \phi_1$ $C_{ij} = \phi_1 C_{i-1,j},\ i > j+1$ $\lvert\phi_1\rvert < 1$	1	2	$1 + \omega$
ar2()	2^{nd} order autoregressive	$C_{ii} = 1,$ $C_{i+1,i} = \phi_1/(1-\phi_2)$ $C_{ij} = \phi_1 C_{i-1,j} + \phi_2 C_{i-2,j},\ i > j+1$ $\lvert\phi_1 \pm \phi_2\rvert < 1,$ $\lvert\phi_1\rvert < 1,\ \lvert\phi_2\rvert < 1$	2	3	$2 + \omega$
ar3()	3^{rd} order autoregressive	$C_{ii} = 1, \Omega = 1 - \phi_2 - \phi_3(\phi_1 + \phi_3),$ $C_{i+1,i} = (\phi_1 + \phi_2\phi_3)/\Omega,$ $C_{i+2,i} = (\phi_1(\phi_1 + \phi_3) + \phi_2(1-\phi_2))/\Omega,$ $C_{ij} = \phi_1 C_{i-1,j} + \phi_2 C_{i-2,j} + \phi_3 C_{i-3,j},\ i > j+2$ $\lvert\phi_1\rvert < (1-\phi_2), \lvert\phi_2\rvert < 1, \lvert\phi_3\rvert < 1$	3	4	$3 + \omega$
sar()	symmetric autoregressive	$C_{ii} = 1,$ $C_{i+1,i} = \phi_1/(1+\phi_1^2/4)$ $C_{ij} = \phi_1 C_{i-1,j} - \phi_1^2/4\ C_{i-2,j},$ $i > j+1$ $\lvert\phi_1\rvert < 1$	1	2	$1 + \omega$
sar2()	constrained autoregressive 3 used for competition	as for `AR3` using $\phi_1 = \gamma_1 + 2\gamma_2,$ $\phi_2 = -\gamma_2(2\gamma_1 + \gamma_2),$ $\phi_3 = \gamma_1\gamma_2^2,$	2	3	$2 + \omega$
ma1()	1^{st} order moving average	$C_{ii} = 1,$ $C_{i+1,i} = -\theta_1/(1+\theta_1^2)$ $C_{ji} = 0,\ j > i+2$ $\lvert\theta_1\rvert < 1$	1	2	$1 + \omega$
ma2()	2^{nd} order moving average	$C_{ii} = 1,$ $C_{i+1,i} = -\theta_1(1-\theta_2)/(1+\theta_1^2+\theta_2^2)$ $C_{i+2,i} = -\theta_2/(1+\theta_1^2+\theta_2^2)$ $C_{ji} = 0,\ j > i+2$ $\theta_2 \pm \theta_1 < 1$ $\lvert\theta_1\rvert < 1,\ \lvert\theta_2\rvert < 1$	2	3	$2 + \omega$

B Available variance models

Details of the available variance models

function name	description	algebraic form	number of parameters		
			corr	hom variance	het variance
arma()	autoregressive moving average	$C_{ii} = 1$, $C_{i+1,i} = (\theta - \phi)(1 - \theta\phi)/(1 + \theta^2 - 2\theta\phi)$ $C_{ji} = \phi C_{j-1,i},\ j > i+1$ $\lvert\theta\rvert < 1,\ \lvert\phi\rvert < 1$	2	3	$2 + \omega$
cor()	uniform correlation	$C_{ii} = 1,\ C_{ij} = \theta,\ i \neq j$	1	2	$1 + \omega$
corb()	banded correlation	$C_{ii} = 1$ $C_{i+j,i} = \phi_j,\ 1 \leq j < \omega$ $\lvert\phi_j\rvert < 1$	j	$j + 1$	$j + \omega$
corg()	general correlation corgh() = us()	$C_{ii} = 1$ $C_{ij} = \phi_{ij},\ i \neq j$ $\lvert\phi_{ij}\rvert < 1$	$\frac{\omega(\omega-1)}{2}$	$\frac{\omega(\omega-1)}{2}+1$	$\frac{\omega(\omega-1)}{2}$ $+ \omega$
One-dimensional unequally spaced power models					
exp()	exponential	$C_{ii} = 1$ $C_{ij} = \phi^{\lvert x_i - x_j\rvert},\ i \neq j$ x_i are *coordinates* $\lvert\phi\rvert < 1$	1	2	$1 + \omega$
gau()	gaussian	$C_{ii} = 1$ $C_{ij} = \phi^{(x_i - x_j)^2}$ x_i are *coordinates* $\lvert\phi\rvert < 1$	1	2	$1 + \omega$
lvr()	linear variance	$C_{ij} = (1 - \theta_{ij})$ $0 < \phi_1$	1	2	$1 + \omega$
Two-dimensional irregularly spaced power models					
iexp()	isotropic exponential	$C_{ii} = 1$ $C_{ij} = \phi^{\lvert x_i - x_j\rvert + \lvert y_i - y_j\rvert}$ $\boldsymbol{x}$ and $\boldsymbol{y}$ *vectors of coordinates* $\lvert\phi\rvert < 1$	1	2	$1 + \omega$
igau()	isotropic gaussian	$C_{ii} = 1$ $C_{ij} = \phi^{(x_i - x_j)^2 + (y_i - y_j)^2}$ $\boldsymbol{x}$ and $\boldsymbol{y}$ *vectors of coordinates* $\lvert\phi\rvert < 1$	1	2	$1 + \omega$

B Available variance models

Details of the available variance models

function name	description	algebraic form	number of parameters		
			corr	hom variance	het variance
ieuc()	isotropic euclidean	$C_{ii} = 1$ $C_{ij} = \phi^{\sqrt{(x_i-x_j)^2+(y_i-y_j)^2}}$ $\boldsymbol{x}$ and $\boldsymbol{y}$ *vectors of coordinates* $\lvert\phi\rvert < 1$	1	2	$1+\omega$
sph()	spherical	$C_{ij} = 1 - \frac{3}{2}\theta_{ij} + \frac{1}{2}\theta_{ij}^3$ $0 < \phi_1$	1	2	$1+\omega$
cir()	circular (Webster and Oliver (2001))	$C_{ij} = 1$ $\quad - \frac{2}{\pi}(\theta_{ij}\sqrt{1-\theta_{ij}^2} + \sin^{-1}\theta_{ij})$ $0 < \phi_1$	1	2	$1+\omega$
aexp()	anisotropic exponential	$C_{ii} = 1$ $C_{ij} = \phi_1^{\lvert x_i-x_j\rvert}\phi_2^{\lvert y_i-y_j\rvert}$ $\boldsymbol{x}$ and $\boldsymbol{y}$ *vectors of coordinates* $\lvert\phi_1\rvert < 1, \lvert\phi_2\rvert < 1$	2	3	$2+\omega$
agau()	anisotropic gaussian	$C_{ii} = 1$ $C_{ij} = \phi_1^{(x_i-x_j)^2}\phi_2^{(y_i-y_j)^2}$ $\boldsymbol{x}$ and $\boldsymbol{y}$ *vectors of coordinates* $\lvert\phi_1\rvert < 1, \lvert\phi_2\rvert < 1$	2	3	$2+\omega$
mtrn()	Matérn with first $1 \le k \le 5$ parameters specified by the user	C_{ij} =Matérn: see Section 4.3.3 $\phi > 0$ range, ν shape(0.5) $\delta > 0$ anisotropy ratio(1), α anisotropy angle(0), $\lambda(1\vert 2)$ metric(2)	k	k+1	$k+\omega$
Heterogeneous variance models					
diag()	diagonal = idh()	$\boldsymbol{\Sigma}_{ii} = \phi_i$ $\boldsymbol{\Sigma}_{ij} = 0,\ i \neq j$	-	-	ω
us()	unstructured general covariance matrix	$\boldsymbol{\Sigma}_{ij} = \phi_{ij}$	-	-	$\frac{\omega(\omega+1)}{2}$
ante(,k)	antedependence order k $1 \le$ *order* $\le \omega - 1$	$\boldsymbol{\Sigma}^{-1} = \boldsymbol{U}\boldsymbol{D}\boldsymbol{U}'$ $D_{ii} = d_i,\ D_{ij} = 0,\ i \neq j$ $U_{ii} = 1,\ U_{ij} = u_{ij},\ 1 \le j-i \le$ order $U_{ij} = 0,\ i > j$	-	-	$(k+1)(\omega - k/2)$

B Available variance models

Details of the available variance models

function name	description	algebraic form	number of parameters		
			corr	hom variance	het variance
chol(,*k*)	cholesky order k $1 \leq order \leq \omega - 1$	$\boldsymbol{\Sigma} = \boldsymbol{LDL'}$ $D_{ii} = d_i,\ D_{ij} = 0,\ i \neq j$ $L_{ii} = 1,\ L_{ij} = l_{ij},\ 1 \leq i-j \leq$ order	-	-	$(k+1)(\omega - k/2)$
sfa(,*k*)	factor analytic order *k*, scaled form	$\boldsymbol{\Sigma} = \boldsymbol{DCD}$, $\boldsymbol{C} = \boldsymbol{FF}^{\mathsf{T}} + \boldsymbol{E}$, $\boldsymbol{F}$ *contains correlation factors* $\boldsymbol{E}$ *diagonal* $\boldsymbol{DD} = \mathrm{d}iag\,(\boldsymbol{\Sigma})$	-	-	$k\omega + \omega$
fa(,*k*)	factor analytic order *k*, sparse form	$\boldsymbol{\Sigma} = \boldsymbol{\Gamma\Gamma'} + \boldsymbol{\Psi}$, $\boldsymbol{\Gamma}$ *contains covariance factors* $\boldsymbol{\Psi}$ *contains specific variance*	-	-	$k\omega + \omega$
facv(,*k*)	factor analytic order *k*, covariance form	$\boldsymbol{\Sigma} = \boldsymbol{\Gamma\Gamma'} + \boldsymbol{\Psi}$, $\boldsymbol{\Gamma}$ *contains covariance factors* $\boldsymbol{\Psi}$ *contains specific variance*	-	-	$k\omega + \omega$
rr(,*k*)	reduced rank order *k*	$\boldsymbol{\Sigma} = \boldsymbol{\Gamma\Gamma'}$ $\boldsymbol{\Gamma}$ *contains covariance factors*	-	-	$k\omega$
Known variance structures					
vm()			-	-	–
General variance models					
str()	variance model relating to a sequence of terms in the model		-	-	–
dsum()	direct sum structures for the residual error term		-	-	–

C 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 C.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 C.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`, E.1	extended functionality
`control`	removed	terminates call	use `asreml.options`, D.5	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()`, E.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`, D.2	more unified framework for related arguments
`na.method.Y`	removed	terminates call	use `na.action`, D.2	
`rcov`	removed	terminates call	`residual`, D.1.1	more appropriate name
`asreml.`*argument*	removed	terminates call	`asr_`*argument*, D.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`, D.2	more appropriate name
`pwrpoints`	deleted	terminates call	`pwr.points`, D.2	more appropriate name
`splinepoints`	deleted	terminates call	`knot.points`, D.2	more appropriate name
`splinescale`	deleted	terminates call	`spline.scale`, D.5	more appropriate name

C 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`, E.2	avoid inconsistent use of `at()`
`giv`	removed	R error will be generated	use `vm()`, E.2	more unified framework for specification of special variance models
`ped`	removed	R error will be generated	use `vm()`, E.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`, D.6	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`, D.6	more unified framework for related components of the `asreml` object
`sparse.formula`	removed	reports `NULL`	use `formulae`, D.6	more unified framework for related components of the `asreml` object
Methods or functions				
`asreml.Ainverse()`	removed	R error will be generated	`ainverse()`, D.5	more appropriate function name
`asreml.control()`	removed	R error will be generated	`asreml.options`, D.5	now a function that groups job control and less frequently used options
`asreml.variogram()`	removed	R error will be generated	use `asr_varioGram`, D.5	renamed to avoid conflict with similar naming in other R packages
`variogram()`	removed	R error will be generated	`varioGram()`, D.5	renamed to avoid conflict with similar naming in other R packages
`variogram.asreml()`	removed	R error will be generated	`varioGram.asreml()`, D.5	renamed to avoid conflict with similar naming in other R packages

D 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.

D.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.

D.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 E.2.

D.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 D.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 09:53:49 2018
          LogLik        Sigma2     DF     wall    cpu
 1      -28.7733           1.0     96 09:53:49    0.0
 2        2.7153           1.0     96 09:53:49    0.0
 3       35.8161           1.0     96 09:53:49    0.0
 4       55.9947           1.0     96 09:53:49    0.0
 5       64.0512           1.0     96 09:53:49    0.0
 6       65.0829           1.0     96 09:53:49    0.0
 7       65.1126           1.0     96 09:53:49    0.0
 8       65.1127           1.0     96 09:53:49    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.

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

D.1 Specification of the linear mixed model

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 09:55:44 2018
          LogLik        Sigma2     DF     wall    cpu
 1       58.2386     0.0608567     96 09:55:44    0.0
 2       60.4027     0.0578118     96 09:55:44    0.0
 3       62.6539     0.0546469     96 09:55:44    0.0
 4       64.1290     0.0524343     96 09:55:44    0.0
 5       64.8815     0.0510398     96 09:55:44    0.0
 6       65.0836     0.0504189     96 09:55:44    0.0
 7       65.1117     0.0501896     96 09:55:44    0.0
 8       65.1127     0.0501448     96 09:55:44    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 09:58:08 2018
          LogLik        Sigma2     DF     wall    cpu
 1       58.2386     0.0608567     96 09:58:08    0.0
 2       60.4027     0.0578118     96 09:58:08    0.0
 3       62.6539     0.0546469     96 09:58:08    0.0
 4       64.1290     0.0524343     96 09:58:08    0.0
 5       64.8815     0.0510398     96 09:58:08    0.0
 6       65.0836     0.0504189     96 09:58:08    0.0
 7       65.1117     0.0501896     96 09:58:08    0.0
 8       65.1127     0.0501448     96 09:58:08    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 E.2 for more detail.

D.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 bi-variate 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 E.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).

D.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.

D.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.

D.5 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:

aipenalty 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 = 5) #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 D.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()`.

D.6 Lists

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

E What's new in ASReml-R Version 4

E.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()` (E.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()` (E.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

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

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 $v_{3,3}$.

E.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 E.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.

`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)
```

E.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.

`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
```

E.3 Methods and functions

```
# 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

Box, G. E. P. (1950). Analysis of growth and wear curves, *Biometrics* **6**: 362–389.

Breslow, N. (2003). Whither PQL?, *Technical Report 192*, UW Biostatistics Working Paper Series, University of Washington.
URL: *http://www.bepress.com/uwbiostat/paper192/*

Breslow, N. E. and Clayton, D. G. (1993). Approximate inference in generalized linear mixed models, *Journal of the American Statistical Association* **88**: 9–25.

Breslow, N. and Lin, X. (1995). Bias correction in generalised linear mixed models with a single component of dispersion, *Biometrika* **82**: 81–91.

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

Cressie, N. A. C. (1991). *Statistics for spatial data*, John Wiley and Sons.

Cullis, B. R. and Gleeson, A. C. (1991). Spatial analysis of field experiments - an extension to two dimensions, *Biometrics* **47**: 1449–1460.

Cullis, B. R., Gleeson, A. C., Lill, W. J., Fisher, J. A. and Read, B. J. (1989). A new procedure for the analysis of early generation variety trials, *Applied Statistics* **38**: 361–375.

Dempster, A. P., Selwyn, M. R., Patel, C. M. and Roth, A. J. (1984). Statistical and computational aspects of mixed model analysis., *Applied Statistics* **33**: 203–214.

Diggle, P. J., Ribeiro, P. J. and Christensen, O. F. (2003). An introduction to model-based geostatistics, *in* J. Moller (ed.), *Spatial Statistics and Computational Methods*, Springer-Verlag, pp. 43–86.

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

Gelfand, A. E., Diggle, P. J., Fuentes, M. and Guttorp, P. (2010). *Handbook of spatial statistics*, Chapman and Hall/CRC.

Gilmour, A. R., Anderson, R. D. and Rae, A. L. (1985). The analysis of binomial data by a generalised linear mixed model, *Biometrika* **72**: 593–599.

Gilmour, A. R., Cullis, B. R. and Verbyla, A. P. (1997). Accounting for natural and extraneous variation in the analysis of field experiments, *Journal of Agricultural, Biological, and Environmental Statistics* **2**: 269–273.

Gilmour, A. R., Cullis, B. R., Welham, S. J., Gogel, B. J. and Thompson, R. (2004). An efficient computing strategy for prediction in mixed linear models, *Computational Statistics and Data Analysis* **44**: 571–586.

Gilmour, A. R., Gogel, B., Cullis, B. R., Welham, S. J. and Thompson, R. (2002). *ASReml User Guide Release 1.0*, VSN International, 5 The Waterhouse, Waterhouse St, Hemel Hempstead, HP1 1ES, UK.

Gilmour, A. R., Thompson, R. and Cullis, B. R. (1995). AI, an efficient algorithm for REML estimation in linear mixed models, *Biometrics* **51**: 1440–1450.

Gleeson, A. C. and Cullis, B. R. (1987). Residual maximum likelihood (REML) estimation of a neighbour model for field experiments, *Biometrics* **43**: 277–288.

Goldstein, H. and Rasbash, J. (1996). Improved approximations for multilevel models with binary response, *Journal of the Royal Statistical Society, Series A* **159**: 505–513.

Goldstein, H., Rasbash, J., Plewis, I., Draper, D., Browne, W., Yang, M., Woodhouse, G. and Healy, M. (1998). *A user's guide to MLwiN*, Institute of Education, London.
URL: *http://multilevel.ioe.ac.uk/*

Green, P. J. and Silverman, B. W. (1994). *Nonparametric regression and generalized linear models*, Chapman and Hall.

Harvey, W. R. (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, W. R. (1977). *Users' guide to LSML76*, Ohio State University, Columbus.

Harville, D. and Mee, R. (1984). A mixed model procedure for analysing ordered categorical data, *Biometrics* **40**: 393–408.

Haskard, K. A. (2006). *Anisotropic Matérn correlation and other issues in model-based geostatistics*, PhD thesis, BiometricsSA, University of Adelaide.

Haskard, K. A., Cullis, B. R. and Verbyla, A. P. (2007). Anisotropic matérn correlation and spatial prediction using reml, *Journal of Agricultural and Biological Sciences* **12**: 147–160.

Kammann, E. E. and Wand, M. P. (2003). Geoadditive models, *Applied Statistics* **52**(1): 1–18.

Keen, A. (1994). Procedure IRREML, *GLW-DLO Procedure Library Manual*, Agricultural Mathematics Group, Wageningen, The Netherlands, pp. Report LWA–94–16.

Kendall, M. G. and Stuart, A. (1969). *The Advanced Theory of Statistics*, Vol. 1, Charles Griffin, London.

Kenward, M. G. and Roger, J. H. (1997). The precision of fixed effects estimates from restricted maximum likelihood, *Biometrics* **53**: 983–997.

Lane, P. W. and Nelder, J. A. (1982). Analysis of covariance and standardisation as instances of predicton, *Biometrics* **38**: 613–621.

McCullagh, P. and Nelder, J. A. (1994). *Generalized Linear Models*, 2 edn, Chapman and Hall, London.

McCulloch, C. and Searle, S. R. (2001). *Generalized, Linear, and Mixed Models*, Wiley.

Meuwissen, T. H. E. and Luo, Z. (1992). Computing inbreeding coefficients in large populations, *Genetics Selection Evolution* **24**.

Millar, R. and Willis, T. (1999). Estimating the relative density of snapper in and around a marine reserve using a log-linear mixed-effects model, *Australian and New Zealand Journal of Statistics* **41**: 383–394.

Nelder, J. A. (1977). A reformulation of linear models, *Journal of the Royal Statistical Society, Series A* **140**: 48–76.

Nelder, J. A. (1994). The statistics of linear models: back to basics, *Statistics and Computing* **4**: 221–234.

Pinheiro, J. C. and Bates, D. M. (2000). *Mixed-Effects Models in S and S-PLUS.*, Springer-Verlaag.

Rodriguez, G. and Goldman, N. (2001). Improved estimation procedures for multilevel models with binary response: A case study, *Journal of the Royal Statistical Society, Series A* **164**(2): 339–355.

Schall, R. (1991). Estimation in generalized linear models with random effects, *Biometrika* **78**(4): 719–27.

Searle, S. R. (1971). *Linear Models*, John Wiley and sons, inc.

Stein, M. L. (1999). *Interpolation of Spatial Data: Some Theory for Kriging*, Springer-Verlag, New York.

Stevens, M. M., Fox, K. M., Warren, G. N., Cullis, B. R., Coombes, N. E. and Lewin, L. G. (1999). An image analysis technique for assessing resistance in rice cultivars to root-feeding chironomid midge larvae (diptera: Chironomidae), *FieldRsct:01a Crops Research* **66**: 25–26.

Stroup, W. W., Baenziger, P. S. and Mulitze, D. K. (1994). Removing spatial variation from wheat yield trials: a comparison of methods, *Crop. Sci* **86**: 62–66.

Thompson, R. (1980). Maximum likelihood estimation of variance components, *Math. Operationsforsch Statistics, Series, Statistics* **11**: 545–561.

Thompson, R., Cullis, B., Smith, A. and Gilmour, A. (2003). A sparse implementation of the average information algorithm for factor analytic and reduced rank variance models, *Australian and New Zealand Journal of Statistics* **45**: 445–459.

Verbyla, A. P., Cullis, B. R., Kenward, M. G. and Welham, S. J. (1999). The analysis of designed experiments and longitudinal data using smoothing splines, *Journal of the Royal Statistical Society, Series C* pp. 269–311.

Waddington, D., Welham, S. J., Gilmour, A. R. and Thompson, R. (1994). Comparisons of some glmm estimators for a simple binomial model., *Genstat Newsletter* **30**: 13–24.

Webster, R. and Oliver, M. A. (2001). *Geostatistics for Environmental Scientists*, John Wiley and Sons, Chichester.

Welham, S. J. (2005). Glmm fits a generalized linear mixed model., *in* R. Payne and P. Lane (eds), *GenStat Reference Manual 3: Procedure Library PL17,*, VSN International, Hemel Hempstead, UK, pp. 260–265.

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

Welham, S. J. and Thompson, R. (1997). Likelihood ratio tests for fixed model terms using residual maximum likelihood., *Journal of the Royal Statistical Society, Series B* **59**: 701–714.

Wolfinger, R. D. (1996). Heterogeneous variance-covariance structures for repeated measures, *Journal of Agricultural, Biological, and Environmental Statistics* **1**: 362–389.

Wolfinger, R. and O'Connell, M. (1993). Generalized linear mixed models: A pseudo-likelihood approach, *Journal of Statistical Computation and Simulation* **48**: 233–243.

Yates, F. (1935). Complex experiments, *Journal of the Royal Statistical Society, Series B* **2**: 181–247.