

**mixed models for S language environments**

## **ASReml-R reference manual**

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

D.G. Butler

B.R. Cullis

A.R. Gilmour

B.J. Gogel

Version 3  
Draft Copy March 2009



**Queensland Government**  
Department of Primary Industries and Fisheries

QE02001

ISSN 0812-0005

*The authors gratefully acknowledge the Grains Research and Development Corporation of Australia for their financial support. We thank the Qld Department of Primary Industries & Fisheries and the NSW Department of Primary Industries for permitting this research to be undertaken and for providing a stimulating environment for applied biometrical consulting and research. We also sincerely thank Robin Thompson, Sue Welham, Ari Verbyla, Alison Smith and Alison Kelly for their contributions and support.*

The Department of Primary Industries and Fisheries (DPI&F) seeks to maximise the economic potential of Queensland's primary industries on a sustainable basis.

© The State of Queensland, Department of Primary Industries and Fisheries, 2002, 2007, 2009.

Except as permitted by the Copyright Act 1968, no part of the work may in any form or by any electronic, mechanical, photocopying, recording, or any other means be reproduced, stored in a retrieval system or be broadcast or transmitted without the prior written permission of DPI&F. The information contained herein is subject to change without notice. The copyright owner shall not be liable for technical or other errors or omissions contained herein. The reader/user accepts all risks and responsibility for losses, damages, costs and other consequences resulting directly or indirectly from using this information.

Enquiries about reproduction, including downloading or printing the web version, should be directed to [ipcu@dpi.qld.gov.au](mailto:ipcu@dpi.qld.gov.au) or telephone +61 7 3225 1398.

---

ASReml is a trademark of NSW Department of Primary Industries, Rothamsted Research, and VSN International Ltd.

R is a trademark of The R Foundation for Statistical Computing.

Windows is a trademark of Microsoft Corporation.

D.G. BUTLER  
Queensland Department of Primary Industries and Fisheries  
PO Box 102  
Toowoomba Qld 4350  
Australia  
david.butler@dpi.qld.gov.au

B. R. CULLIS  
NSW Department of Primary Industries  
Wagga Wagga  
Australia  
brian.cullis@dpi.nsw.gov.au

A. R. GILMOUR  
NSW Department of Primary Industries  
Orange  
Australia  
arthur.gilmour@dpi.nsw.gov.au

B. J. GOGEL  
University of Adelaide  
PO Box xxx  
Adelaide SA xxxx  
Australia  
beverley.gogel@unisa.edu.au

**Mixed Models for S language environments**

**ASReml-R reference manual**

Training Series QE02001

Queensland Department of Primary Industries and Fisheries  
NSW Department of Primary Industries

# Preface

ASReml-R fits the linear mixed model using Residual Maximum Likelihood (REML) and is a joint venture between the Queensland Department of Primary Industries & Fisheries (QDPI&F) and the Biometrics Program of the NSW Department of Primary Industries. ASReml-R uses the numerical routines from the standalone program ASReml™ [Gilmour et al., 2002], under joint development through the NSW Department of Primary Industries the Biomathematics and Bioinformatics Division of Rothamsted Research. This guide describes Version 3.00 of ASReml-R, released in March 2009.

Linear mixed effects models provide a rich and flexible tool for the analysis of many datasets commonly arising in the agricultural, biological, medical and environmental sciences. Typical applications include the analysis of balanced and unbalanced longitudinal data, repeated measures, balanced and unbalanced designed experiments, multi-environment trials, multivariate datasets and regular or irregular spatial data.

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. ASReml-R requires that a dynamic link library (Microsoft Windows™) or shared object file (Linux) containing the numerical methods be loaded at runtime.

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 datasets.

This manual consists of nine 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; Chapter 4 is a key chapter that presents the syntax for specifying variance models for random effects in the model; Chapter 3.13 describes the model specification for a multivariate analyses; 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; Chapter 7 describes the ASReml-R class and related methods and finally Chapter 8 presents a comprehensive and diverse set of worked examples.

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 and the shared object library are being continually upgraded to implement new developments in the application of linear mixed models. The release version will be distributed on CD to licensed users while a developmental version (and fixes) will be available to licensees from <http://www.vsni.co.uk>.

# Contents

<b>Preface</b>	<b>i</b>
<b>Contents</b>	<b>iii</b>
<b>List of Tables</b>	<b>vii</b>
<b>List of Figures</b>	<b>viii</b>
<b>1 Introduction</b>	<b>1</b>
1.1 What ASReml-R can do . . . . .	1
1.2 Getting started . . . . .	1
1.2.1 Installation . . . . .	1
1.2.2 Help and references . . . . .	2
1.2.3 Conventions . . . . .	2
1.2.4 Using this guide . . . . .	2
1.3 Data sets used . . . . .	3
1.3.1 Nebraska Intrastate Nursery (NIN) field experiment . . . . .	3
1.3.2 Repeated measures on rats . . . . .	3
1.3.3 Orange wether trial . . . . .	3
1.3.4 Beef cattle data . . . . .	5
<b>2 Some theory</b>	<b>8</b>
2.1 The linear mixed model . . . . .	8
2.1.1 Introduction . . . . .	8
2.1.2 Direct product structures . . . . .	8
2.1.3 Variance structures for the errors: R structures . . . . .	10
2.1.4 Variance structures for the random effects: G structures . . . . .	11
2.2 Estimation . . . . .	12
2.2.1 Variance parameters . . . . .	12
2.2.2 Fixed and Random effects . . . . .	14
2.3 What are BLUPs? . . . . .	15
2.4 Combining variance models . . . . .	15
2.5 Inference for random effects . . . . .	16
2.5.1 Tests of hypotheses . . . . .	16
2.5.2 Diagnostics . . . . .	17

2.6	Inference for fixed effects . . . . .	18
2.6.1	Incremental and Conditional Wald Statistics . . . . .	19
2.7	Kenward and Roger Adjustments . . . . .	21
2.8	Approximate stratum variances . . . . .	22
<b>3</b>	<b>Fitting the mixed model</b>	<b>23</b>
3.1	The data frame . . . . .	23
3.2	Introducing the <code>asreml()</code> function call . . . . .	25
3.2.1	Model formulae: specifying the linear mixed model . . . . .	25
3.2.2	Finding the data . . . . .	26
3.3	Components of the fitted model: the <code>asreml</code> object . . . . .	26
3.3.1	Methods and related functions . . . . .	26
3.4	A note on data order . . . . .	27
3.5	Getting help . . . . .	27
3.6	Fixed terms . . . . .	27
3.6.1	Dense fixed terms . . . . .	27
3.6.2	Sparse fixed terms . . . . .	30
3.6.3	Covariates . . . . .	30
3.7	Random terms . . . . .	30
3.7.1	Initial values and constraints for variance parameters . . . . .	30
3.7.2	Specifying variance structures . . . . .	32
3.8	Conditional factors: the <code>at()</code> function . . . . .	32
3.9	Weights . . . . .	32
3.10	Missing values . . . . .	33
3.10.1	Missing values in the response . . . . .	33
3.10.2	Missing values in the explanatory variables . . . . .	33
3.11	Generalized linear models . . . . .	33
3.12	Generalized Linear Mixed Models . . . . .	34
3.13	Multivariate analysis . . . . .	35
3.13.1	Model specification . . . . .	35
3.13.2	Specifying multivariate variance structures . . . . .	37
3.14	Testing of terms: the <code>wald()</code> method . . . . .	38
<b>4</b>	<b>Specifying variance structures</b>	<b>39</b>
4.1	A sequence of structures for the NIN field trial data . . . . .	40
4.2	Types of variance models . . . . .	43
4.2.1	Correlation models . . . . .	43
4.2.2	Homogeneous variance models . . . . .	44
4.2.3	Heterogeneous variance models . . . . .	44
4.2.4	Positive definite matrices . . . . .	45
4.3	Variance model functions . . . . .	45
4.3.1	Default identity . . . . .	45
4.3.2	Time series type models . . . . .	46
4.3.3	Metric based models in $\mathcal{R}$ or $\mathcal{R}^2$ . . . . .	47
4.3.4	General structure models . . . . .	50
4.3.5	Known relationship structures . . . . .	53
4.3.6	General variance structures . . . . .	54
4.4	Default initial values for variance parameters . . . . .	54

4.5	Rules for combining variance models . . . . .	54
4.6	Constraining variance parameters . . . . .	55
<b>5</b>	<b>Genetic analysis</b>	<b>58</b>
5.1	Introduction . . . . .	58
5.2	Pedigree, G-inverse objects and genetic groups . . . . .	58
5.2.1	Pedigree objects . . . . .	58
5.2.2	giv objects . . . . .	59
5.2.3	Genetic groups . . . . .	60
5.3	Generating an A-inverse matrix with <code>asreml.Ainverse()</code> . . . . .	61
5.4	Using Pedigree and G-inverse objects . . . . .	61
<b>6</b>	<b>Prediction from the linear model</b>	<b>63</b>
6.1	Introduction . . . . .	63
6.2	The predict method . . . . .	64
6.2.1	The prediction process . . . . .	65
6.3	Aliasing . . . . .	66
6.4	Complicated weighting . . . . .	67
6.5	Further examples . . . . .	68
<b>7</b>	<b>The <code>asreml</code> class</b>	<b>70</b>
7.1	<code>asreml</code> . . . . .	70
	<code>asreml</code> . . . . .	70
7.2	<code>asreml.control</code> . . . . .	73
	<code>asreml.control</code> . . . . .	74
7.3	The <code>asreml</code> object . . . . .	76
	<code>asreml.object</code> . . . . .	76
7.4	Methods and related functions . . . . .	77
7.4.1	<code>asreml.About</code> . . . . .	77
7.4.2	<code>asreml.Ainverse</code> . . . . .	78
7.4.3	<code>asreml.asUnivariate</code> . . . . .	79
7.4.4	<code>asreml.constraints</code> . . . . .	80
7.4.5	<code>asreml.man</code> . . . . .	81
7.4.6	<code>asreml.read.table</code> . . . . .	81
7.4.7	<code>asreml.variogram</code> . . . . .	82
7.4.8	<code>coef.asreml</code> . . . . .	83
7.4.9	<code>fitted.asreml</code> . . . . .	84
7.4.10	<code>plot.asreml</code> . . . . .	84
7.4.11	<code>plot.asrVariogram</code> . . . . .	85
7.4.12	<code>predict.asreml</code> . . . . .	86
7.4.13	<code>residuals.asreml</code> . . . . .	88
7.4.14	<code>summary.asreml</code> . . . . .	88
7.4.15	<code>svc.asreml</code> . . . . .	89
7.4.16	<code>tr.asreml</code> . . . . .	90
7.4.17	<code>update.asreml</code> . . . . .	90
7.4.18	<code>variogram.asreml</code> . . . . .	91
7.4.19	<code>wald.asreml</code> . . . . .	92
	<code>wald.asreml</code> . . . . .	93

<b>8</b>	<b>Examples</b>	<b>95</b>
8.1	Introduction . . . . .	95
8.2	Split Plot Design . . . . .	95
8.3	Unbalanced nested design . . . . .	99
8.4	Sources of variability in unbalanced data . . . . .	102
8.5	Balanced repeated measures . . . . .	104
8.6	Spatial analysis of a field experiment . . . . .	110
8.7	Unreplicated early generation variety trial . . . . .	115
8.8	Paired Case-Control Study . . . . .	119
8.9	Balanced longitudinal data . . . . .	129
<b>A</b>	<b>Some technical details about model fitting in <code>asreml()</code></b>	<b>135</b>
A.1	Sparse <i>versus</i> dense . . . . .	135
A.2	Ordering of terms in <code>asreml()</code> . . . . .	135
A.3	Aliasing and singularities . . . . .	135
A.3.1	Examples of aliasing . . . . .	136
<b>B</b>	<b>Available variance models</b>	<b>137</b>
	<b>Bibliography</b>	<b>142</b>
	<b>Index</b>	<b>146</b>

# List of Tables

1.1	Trial layout and allocation of varieties to plots in the NIN field trial . . .	4
2.1	Combination of G and R structures . . . . .	16
3.1	Summary of reserved names . . . . .	28
3.2	Families and link functions . . . . .	34
4.1	Sequence of variance structures for the NIN field trial . . . . .	44
8.1	A split-plot field trial of oat varieties and nitrogen application . . . . .	95
8.2	Rat data: ANOVA decomposition . . . . .	100
8.3	REML log-likelihood test for the voltage data . . . . .	104
8.4	Summary of variance models fitted to the plant data . . . . .	106
8.5	Summary of Wald statistics for the plant data . . . . .	110
8.6	Field layout of Slate Hall Farm experiment . . . . .	111
8.7	Summary of models fitted to the Slate Hall data . . . . .	114
8.8	Estimated components from univariate analyses of bloodworm data . . .	124
8.9	Equivalence of random effects in bivariate and univariate analyses . . .	125
8.10	Estimated components from bivariate analysis of bloodworm data . . .	125
8.11	ANOVA decomposition for the orange data . . . . .	131
8.12	Sequence of models fitted to the <b>orange</b> data . . . . .	132
A.1	Examples of aliasing . . . . .	136
B.1	Details of the available variance models . . . . .	138

# List of Figures

1.1	Weekly body weights of rats . . . . .	5
8.1	Residual plot for the rat data . . . . .	101
8.2	Residuals vs fitted values for the voltage data . . . . .	103
8.3	Trellis plot of plant height for each of 14 plants . . . . .	105
8.4	<code>residual ~ Plant   Time</code> for the plant data . . . . .	108
8.5	Sample variogram of the AR1×AR1 model for the Slate Hall data . . .	112
8.6	Sample variogram of the AR1×AR1 model for the Tullibigeal data . . .	116
8.7	Sample variogram including <code>pol(column,-1)</code> for the Tullibigeal data . .	118
8.8	Plot of square root of root weight for the bloodworm data . . . . .	121
8.9	BLUPs for treated plotted against BLUPs for control . . . . .	127
8.10	Estimated deviations from regression for the bloodworm data . . . . .	128
8.11	Estimated difference between control and treated for the bloodworm data	128
8.12	Plot of trunk circumference against age for the orange data . . . . .	130
8.13	Fitted cubic smoothing spline for tree 1 . . . . .	131
8.14	Plot of fitted cubic smoothing spline for model 1 . . . . .	133
8.15	Fitted values adjusted for <code>Season</code> for model 6 . . . . .	134

# Introduction

# 1

## 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

Production versions of ASReml-R are available for several implementations on Microsoft Windows and Linux systems. Installation varies with each system and instructions are contained in a separate document distributed with the archive or available from the web site. If the instructions are inadequate then please contact [support@asreml.co.uk](mailto:support@asreml.co.uk) for assistance.

### 1.2.2 Help and references

**Documentation** 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. The function `asreml.man()` displays a copy of this manual in PDF form.

**Forum** There is an ASReml forum that all users are encouraged to join; visit [www.vsni.co.uk/forum](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 (R) and non-residual (random,  $G$ ) levels, will appear in detail in a forthcoming publication.

### 1.2.3 Conventions

This manual uses the following typographic conventions:

`this font` is used to denote operating system commands;

*this font* is used to indicate user supplied arguments to operating system commands, including filenames.

**this font** is used for ASReml-R function examples; `this font` for other R functions and their associated arguments,

**this font** is used for emphasis and user supplied variables to R functions,

`this font` is used for verbatim output of R function calls.

The R command prompt is denoted by `>` and the operating system prompt by `%`.

### 1.2.4 Using this guide

**Introduction** 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

**Theory** values for variance components. An introduction to the theory that underpins the methods in ASReml-R is covered in Chapter 2

**Variance modeling** 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

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

**Function reference** The `asreml()` function and `asreml` class methods are documented in Chapter 7.

**Examples** Chapter 8 presents a wide range of additional worked examples.

## 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**.

### 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.

### 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 file `wether.dat` shown below contains greasy fleece weight (kg), yield (percentage of clean fleece weight to greasy fleece weight) and fibre diameter (microns).

An extract of `orange.csv` is given below:

```
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
```

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

row	1	2	3	4	5	6	7	8	9	10	11
	column										
1	-	NE83407	BUCKSKIN	NE87612	VONA	NE87512	NE87408	CODY	BUCKSKIN	NE87612	KS831374
2	-	CENTURA	NE86527	NE87613	NE87463	NE88407	NE83407	NE87612	NE83406	BUCKSKIN	NE86482
3	-	SCOUT66	NE86582	NE87615	NE86507	NE87403	NORKAN	NE87457	NE87409	NE85556	NE86623
4	-	COLT	NE86606	NE87619	BUCKSKIN	NE87457	REDLAND	NE84557	NE87499	BRULE	NE86527
5	-	NE83498	NE86607	NE87627	ROUGHRRIDER	NE83406	KS831374	NE83712	CENTURA	NE86507	NE87451
6	-	NE84557	ROUGHRRIDER	-	NE86527	COLT	COLT	NE86507	NE83432	ROUGHRRIDER	NE87409
7	-	NE83432	VONA	CENTURA	SCOUT66	NE87522	NE86527	TAM200	NE87512	VONA	GAGE
8	-	NE85556	SIouxLAND	NE85623	NE86509	NORKAN	VONA	NE87613	ROUGHRRIDER	NE83404	NE83407
9	-	NE85623	GAGE	CODY	NE86606	NE87615	TAM107	ARAPAHOE	NE83498	CODY	NE87615
10	-	CENTURAK78	NE83712	NE86582	NE84557	NE85556	CENTURAK78	SCOUT66	-	NE87463	ARAPAHOE
11	-	NORKAN	NE867666	NE87408	KS831374	TAM200	NE87627	NE87403	NE86T666	NE86582	CHEYENNE
12	-	KS831374	NE87403	NE87451	GAGE	LANCOTA	NE867666	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	NE83712	CHEYENNE	BRULE	NE87522	HOMESTEAD	CENTURA	NE87513	NE83498
16	LANCER	LANCOTA	NE87451	NE87409	NE86607	NE87612	CHEYENNE	NE83404	NE86503	NE83712	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

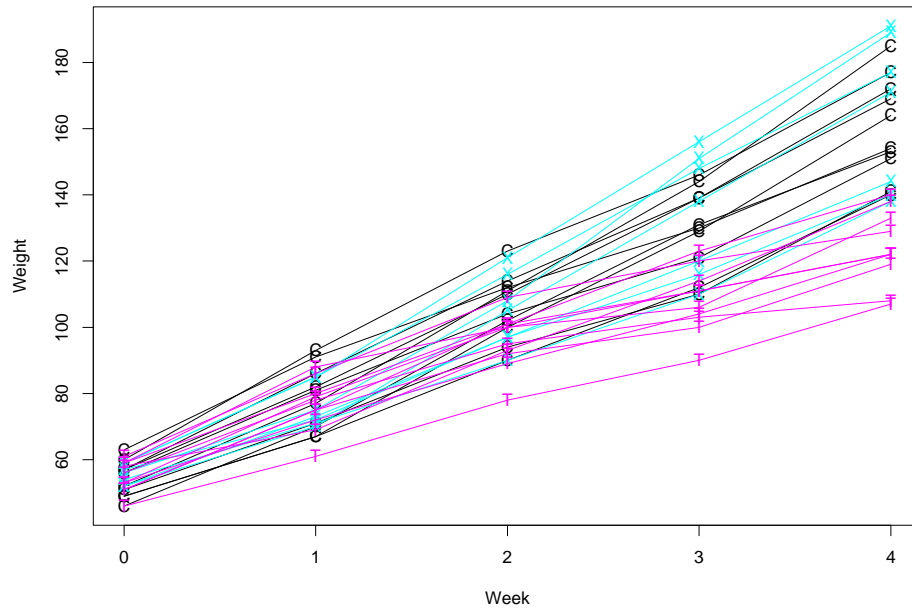


Figure 1.1: Weekly body weights of rats. C = Control, X = Thyroxin, T = Thiouracil

```

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  $y_1$ ,  $y_2$  and  $y_3$ , 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 line 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* .

# Some theory

# 2

## 2.1 The linear mixed model

### 2.1.1 Introduction

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

$$\mathbf{y} = \mathbf{X}\boldsymbol{\tau} + \mathbf{Z}\mathbf{u} + \mathbf{e} \quad (2.1)$$

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

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

$$\begin{bmatrix} \mathbf{u} \\ \mathbf{e} \end{bmatrix} \sim N \left( \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \end{bmatrix}, \theta \begin{bmatrix} \mathbf{G}(\boldsymbol{\gamma}) & \mathbf{0} \\ \mathbf{0} & \mathbf{R}(\boldsymbol{\phi}) \end{bmatrix} \right) \quad (2.2)$$

where the matrices  $\mathbf{G}$  and  $\mathbf{R}$  are functions of parameters  $\boldsymbol{\gamma}$  and  $\boldsymbol{\phi}$ , respectively. The parameter  $\theta$  is a variance parameter which we will refer to as the scale parameter. In mixed effects models with more than one residual variance, arising for example in the analysis of data with more than one section (see below) or variate, the parameter  $\theta$  is fixed to one. In mixed effects models with a single residual variance then  $\theta$  is equal to the residual variance ( $\sigma^2$ ). In this case  $\mathbf{R}$  *must* be correlation matrix (see Table 2.1 for a discussion).

### 2.1.2 Direct product structures

To undertake variance modelling in `asremI()` it is important to understand the formation of variance structures via direct products ( $\otimes$ ). The direct product of two matrices  $\mathbf{A}^{(m \times p)}$  and  $\mathbf{B}^{(n \times q)}$  is

$$\begin{bmatrix} a_{11}\mathbf{B} & \dots & a_{1p}\mathbf{B} \\ \vdots & \ddots & \vdots \\ a_{m1}\mathbf{B} & \dots & a_{mp}\mathbf{B} \end{bmatrix}$$

### Direct products in R structures

Consider a vector of common errors associated with an experiment. The usual least squares assumption (and the default in `asreml()`) is that these are independently and identically distributed (IID). However, if the data was from a field experiment laid out in a rectangular array of  $r$  rows by  $c$  columns, say, we could arrange the residuals  $\mathbf{e}$  as a matrix and potentially 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  $\rho_r$ ) and column model (order  $c$ , autocorrelation parameter  $\rho_c$ ) respectively. More specifically, a two-dimensional separable autoregressive spatial structure ( $\text{AR1} \otimes \text{AR1}$ ) 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

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

See 3.13 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

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

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

### Direct products in G structures

Likewise, the random terms in  $\mathbf{u}$  in the model 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 model term *site.variety* may have the variance structure

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

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. **Note:** 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.



### 2.1.3 Variance structures for the errors: $\mathbf{R}$ structures

The vector  $\mathbf{e}$  will in some situations be a series of vectors indexed by a factor or factors. The convention we adopt is to refer to these as *sections*. Thus  $\mathbf{e} = [e'_1, e'_2, \dots, e'_s]'$  and the  $e_j$  represent the errors of *sections* of the data. For example, these sections may represent different experiments in a multi-environment trial (MET), or different trials in a meta analysis. It is assumed that

$$\mathbf{R} = \oplus_{j=1}^s \mathbf{R}_j = \begin{bmatrix} \mathbf{R}_1 & 0 & \dots & 0 & 0 \\ 0 & \mathbf{R}_2 & \dots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & \mathbf{R}_{s-1} & 0 \\ 0 & 0 & \dots & 0 & \mathbf{R}_s \end{bmatrix}$$

so that each section has its own variance matrix but they are assumed independent.

Cullis et al. [1997] consider the spatial analysis of multi-environment trials in which

$$\begin{aligned} \mathbf{R}_j &= \mathbf{R}_j(\phi_j) \\ &= \sigma_j^2(\boldsymbol{\Sigma}_j(\boldsymbol{\rho}_j) + \psi_j \mathbf{I}_{n_j}) \end{aligned}$$

and each section represents a trial. This model accounts for between trial error variance heterogeneity ( $\sigma_j^2$ ) and possibly a different spatial variance model for each trial.

In the simplest case the matrix  $\mathbf{R}$  could be known and proportional to an identity matrix. Each component matrix,  $\mathbf{R}_j$  (or  $\mathbf{R}$  itself for one section) is assumed to be the kronecker (direct) product of one, two or three component matrices. The component matrices are related to the underlying structure of the data. If the structure is defined by factors, for example, replicates, rows and columns, then the matrix  $\mathbf{R}$  can be constructed as a kronecker product of three matrices describing the nature of the correlation across replicates, rows and columns. These factors must completely describe the structure of the data, which means that

1. the number of combined levels of the factors must equal the number of data points,
2. each factor combination must uniquely specify a single data point.

These conditions are necessary to ensure the expression  $\text{var}(\mathbf{e}) = \theta \mathbf{R}$  is valid. The assumption that the overall variance structure can be constructed as a direct product of matrices corresponding to underlying factors is called the assumption of separability and assumes that any correlation process across levels of a factor is independent of any other factors in the term. This assumption is required to make the estimation process computationally feasible, though it can be relaxed, for certain applications, for example fitting isotropic covariance models to irregularly spaced spatial data. Multivariate data and repeated measures data usually satisfy the assumption of separability. In particular, if the data are indexed by factors **units** and **traits** (for multivariate data) or **times** (for repeated measures data), then the  $\mathbf{R}$  structure may be written as  $\text{units} \otimes \text{traits}$  or  $\text{units} \otimes \text{times}$ .

### 2.1.4 Variance structures for the random effects: $\mathbf{G}$ structures

The  $q \times 1$  vector of random effects is often composed of  $b$  subvectors  $\mathbf{u} = [\mathbf{u}'_1 \ \mathbf{u}'_2 \ \dots \ \mathbf{u}'_b]'$  where the subvectors  $\mathbf{u}_i$  are of length  $q_i$  and these subvectors are usually assumed independent normally distributed with variance matrices  $\theta \mathbf{G}_i$ . Thus just like  $\mathbf{R}$  we have

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

There is a corresponding partition in  $\mathbf{Z}$ ,  $\mathbf{Z} = [\mathbf{Z}_1 \ \mathbf{Z}_2 \ \dots \ \mathbf{Z}_b]$ . As before each submatrix,  $\mathbf{G}_i$ , is assumed to be the kronecker product of one, two or three component matrices. These matrices are indexed for each of the factors constituting the term in the linear model. For example, the term *site:genotype* has two factors and so the matrix  $\mathbf{G}_i$  is comprised of two component matrices defining the variance structure for each factor in the term.

Models for the component matrices  $\mathbf{G}_i$  include the standard model for which  $\mathbf{G}_i = \gamma_i \mathbf{I}_{q_i}$  as well as direct product models for correlated random factors given by

$$\mathbf{G}_i = \mathbf{G}_{i1} \otimes \mathbf{G}_{i2} \otimes \mathbf{G}_{i3}$$

for three component factors. The vector  $\mathbf{u}_i$  is therefore assumed to be the vector representation of a 3-way array. For two factors the vector  $\mathbf{u}_i$  is simply the vec of a matrix with rows and columns indexed by the component factors in the term, where vec of a matrix is a function which stacks the columns of its matrix argument below each other.

A range of models are available for the components of both  $\mathbf{R}$  and  $\mathbf{G}$ . They include correlation ( $C$ ) models (that is, where the diagonals are 1), or covariance ( $V$ ) models and are discussed in detail in Chapter 4 (see Section 4.2). Some correlation models include

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

Some of the covariance models include

- diagonal (that is, independent with heterogeneous variances)
- antedependence

- unstructured
- factor analytic.

There is the facility within `asrem()` to allow for a nonzero covariance between the subvectors of  $\mathbf{u}$ , for example in random regression models. In this setting the intercept and say the slope for each unit are assumed to be correlated and it is more natural to consider the two component terms as a single term, which gives rise to a single G structure. This concept is discussed later.

## 2.2 Estimation

Estimation involves two processes that are very strongly linked. One process involves estimation of  $\boldsymbol{\tau}$  and prediction of  $\mathbf{u}$  (although the latter may not always be of interest) for given  $\theta$ ,  $\phi$  and  $\gamma$ . The other process involves estimation of these variance parameters. Note that in the following sections we have set  $\theta = 1$  to simplify the presentation of results.

### 2.2.1 Variance parameters

Estimation of the variance parameters is carried out using residual or restricted maximum likelihood (REML), developed by Patterson and Thompson [1971]. Note firstly that

$$\mathbf{y} \sim N(\mathbf{X}\boldsymbol{\tau}, \mathbf{H}). \quad (2.3)$$

where  $\mathbf{H} = \mathbf{R} + \mathbf{ZGZ}'$ . REML does not use (2.3) 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  $\mathbf{y}$  using a non-singular matrix  $\mathbf{L} = [\mathbf{L}_1 \ \mathbf{L}_2]$  such that

$$\mathbf{L}'_1 \mathbf{X} = \mathbf{I}_p, \quad \mathbf{L}'_2 \mathbf{X} = \mathbf{0}.$$

If  $\mathbf{y}_j = \mathbf{L}'_j \mathbf{y}$ ,  $j = 1, 2$ ,

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

The full distribution of  $\mathbf{L}'\mathbf{y}$  can be partitioned into a *conditional distribution*, namely  $\mathbf{y}_1|\mathbf{y}_2$ , for estimation of  $\boldsymbol{\tau}$ , and a *marginal distribution* based on  $\mathbf{y}_2$  for estimation of  $\gamma$  and  $\phi$ ; the latter is the basis of the **residual likelihood**.

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

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

Estimation of  $\boldsymbol{\kappa} = [\boldsymbol{\gamma}' \ \boldsymbol{\phi}']'$  is based on the distribution of  $\mathbf{y}_2$ ,

$$\begin{aligned} \ell_R &= -\frac{1}{2}(\log \det \mathbf{L}'_2 \mathbf{H}^{-1} \mathbf{L}_2 + \mathbf{y}'_2 (\mathbf{L}'_2 \mathbf{H} \mathbf{L}_2)^{-1} \mathbf{y}_2) \\ &= -\frac{1}{2}(\log \det \mathbf{X}'\mathbf{H}^{-1}\mathbf{X} + \log \det \mathbf{H} + \mathbf{y}'\mathbf{P}\mathbf{y}) \end{aligned} \quad (2.4)$$

where

$$\mathbf{P} = \mathbf{H}^{-1} - \mathbf{H}^{-1}\mathbf{X}(\mathbf{X}'\mathbf{H}^{-1}\mathbf{X})^{-1}\mathbf{X}'\mathbf{H}^{-1}.$$

Note that  $\mathbf{y}'\mathbf{P}\mathbf{y} = (\mathbf{y} - \mathbf{X}\hat{\boldsymbol{\tau}})'\mathbf{H}^{-1}(\mathbf{y} - \mathbf{X}\hat{\boldsymbol{\tau}})$ . The log-likelihood (2.4) depends on  $\mathbf{X}$  and not on the particular non-unique transformation defined by  $\mathbf{L}$ .

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

$$\ell_R = -\frac{1}{2}(\log \det \mathbf{C} + \log \det \mathbf{R} + \log \det \mathbf{G} + \mathbf{y}'\mathbf{P}\mathbf{y}). \quad (2.5)$$

We can also write

$$\mathbf{P} = \mathbf{R}^{-1} - \mathbf{R}^{-1}\mathbf{W}\mathbf{C}^{-1}\mathbf{W}'\mathbf{R}^{-1}$$

with  $\mathbf{W} = [\mathbf{X} \ \mathbf{Z}]$ . Letting  $\boldsymbol{\kappa} = (\boldsymbol{\gamma}, \boldsymbol{\phi})$ , the REML estimates of  $\kappa_i$  are found by calculating the score

$$U(\kappa_i) = \partial \ell_R / \partial \kappa_i = -\frac{1}{2}[\text{tr}(\mathbf{P}\mathbf{H}_i) - \mathbf{y}'\mathbf{P}\mathbf{H}_i\mathbf{P}\mathbf{y}] \quad (2.6)$$

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

The elements of the observed information matrix are

$$\begin{aligned} -\frac{\partial^2 \ell_R}{\partial \kappa_i \partial \kappa_j} &= \frac{1}{2} \text{tr}(\mathbf{P}\mathbf{H}_{ij}) - \frac{1}{2} \text{tr}(\mathbf{P}\mathbf{H}_i\mathbf{P}\mathbf{H}_j) \\ &\quad + \mathbf{y}'\mathbf{P}\mathbf{H}_i\mathbf{P}\mathbf{H}_j\mathbf{P}\mathbf{y} - \frac{1}{2} \mathbf{y}'\mathbf{P}\mathbf{H}_{ij}\mathbf{P}\mathbf{y} \end{aligned} \quad (2.7)$$

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

The elements of the expected information matrix are

$$\mathbb{E} \left( -\frac{\partial^2 \ell_R}{\partial \kappa_i \partial \kappa_j} \right) = \frac{1}{2} \text{tr}(\mathbf{P}\mathbf{H}_i\mathbf{P}\mathbf{H}_j). \quad (2.8)$$

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

$$\boldsymbol{\kappa}^{(1)} = \boldsymbol{\kappa}^{(0)} + \mathbf{I}(\boldsymbol{\kappa}^{(0)}, \boldsymbol{\kappa}^{(0)})^{-1} \mathbf{U}(\boldsymbol{\kappa}^{(0)}) \quad (2.9)$$

where  $\mathbf{U}(\boldsymbol{\kappa}^{(0)})$  is the score vector (2.6) and  $\mathbf{I}(\boldsymbol{\kappa}^{(0)}, \boldsymbol{\kappa}^{(0)})$  is the expected information matrix (2.8) of  $\boldsymbol{\kappa}$  evaluated at  $\boldsymbol{\kappa}^{(0)}$ .

For large models or large data sets, the evaluation of the trace terms in either (2.7) or (2.8) is either not feasible or is very computer intensive. To overcome this problem the AI algorithm [Gilmour et al., 1995] is used. The matrix denoted by  $\mathcal{I}_A$  is obtained by averaging (2.7) and (2.8) and approximating  $\mathbf{y}'\mathbf{P}\mathbf{H}_{ij}\mathbf{P}\mathbf{y}$  by its expectation,  $\text{tr}(\mathbf{P}\mathbf{H}_{ij})$  in those cases when  $\mathbf{H}_{ij} \neq 0$ . For variance components models (that is, those linear with respect to variances in  $\mathbf{H}$ ), the terms in  $\mathcal{I}_A$  are exact averages of those in (2.7) and (2.8). The basic idea is to use  $\mathcal{I}_A(\kappa_i, \kappa_j)$  in place of the expected information matrix in (2.9) to update  $\boldsymbol{\kappa}$ .

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

$$\mathcal{I}_A(\kappa_i, \kappa_j) = \frac{1}{2} \mathbf{y}'\mathbf{P}\mathbf{H}_i\mathbf{P}\mathbf{H}_j\mathbf{P}\mathbf{y}. \quad (2.10)$$

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

$$\mathbf{y} = [\mathbf{y}_0, \mathbf{y}_1, \dots, \mathbf{y}_k]$$

where  $\mathbf{y}_i, i > 0$  is the ‘working’ variate for  $\kappa_i$  and is given by

$$\begin{aligned} \mathbf{y}_i &= \mathbf{H}_i \mathbf{P} \mathbf{y} \\ &= \mathbf{H}_i \mathbf{R}^{-1} \tilde{\mathbf{e}} \\ &= \mathbf{R}_i \mathbf{R}^{-1} \tilde{\mathbf{e}}, \quad \kappa_i \in \phi \\ &= \mathbf{Z} \mathbf{G}_i \mathbf{G}^{-1} \tilde{\mathbf{u}}, \quad \kappa_i \in \gamma \end{aligned}$$

where  $\tilde{\mathbf{e}} = \mathbf{y} - \mathbf{X} \hat{\boldsymbol{\tau}} - \mathbf{Z} \tilde{\mathbf{u}}$ ,  $\hat{\boldsymbol{\tau}}$  and  $\tilde{\mathbf{u}}$  are solutions to (2.11) and  $\mathbf{y}_0 = \mathbf{y}$ , the data vector. 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.

One process involves estimation of  $\boldsymbol{\tau}$  and prediction of  $\mathbf{u}$  (although the latter may not always be of interest) for given  $\theta$ ,  $\phi$  and  $\gamma$ . The other process involves estimation of these variance parameters.

### 2.2.2 Fixed and Random effects

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

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

is used. This is the log-joint distribution of  $(\mathbf{Y}, \mathbf{u})$ . *It is not a log-likelihood though in extensions to non-normal data it has been treated as a log-likelihood.*

Differentiating with respect to  $\boldsymbol{\tau}$  and  $\mathbf{u}$  leads to the mixed model equations [Robinson, 1991] which are given by

$$\begin{bmatrix} \mathbf{X}' \mathbf{R}^{-1} \mathbf{X} & \mathbf{X}' \mathbf{R}^{-1} \mathbf{Z} \\ \mathbf{Z}' \mathbf{R}^{-1} \mathbf{X} & \mathbf{Z}' \mathbf{R}^{-1} \mathbf{Z} + \mathbf{G}^{-1} \end{bmatrix} \begin{bmatrix} \hat{\boldsymbol{\tau}} \\ \tilde{\mathbf{u}} \end{bmatrix} = \begin{bmatrix} \mathbf{X}' \mathbf{R}^{-1} \mathbf{y} \\ \mathbf{Z}' \mathbf{R}^{-1} \mathbf{y} \end{bmatrix}. \quad (2.11)$$

These can be written as

$$\mathbf{C} \tilde{\boldsymbol{\beta}} = \mathbf{W} \mathbf{R}^{-1} \mathbf{y}$$

where  $\mathbf{C} = \mathbf{W}' \mathbf{R}^{-1} \mathbf{W} + \mathbf{G}^*$ ,  $\mathbf{W} = [\mathbf{X} \quad \mathbf{Z}]$ ,  $\boldsymbol{\beta} = [\boldsymbol{\tau}' \quad \mathbf{u}']'$  and

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

The solution of (2.11) requires values for  $\gamma$  and  $\phi$ . In practice we replace  $\gamma$  and  $\phi$  by their REML estimates  $\hat{\gamma}$  and  $\hat{\phi}$ .

Note that  $\hat{\boldsymbol{\tau}}$  is the best linear unbiased estimator (BLUE) of  $\boldsymbol{\tau}$ , while  $\tilde{\mathbf{u}}$  is the best linear unbiased predictor (BLUP) of  $\mathbf{u}$ , for known  $\gamma$  and  $\phi$ . We also note that

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

## 2.3 What are BLUPs?

Consider a balanced one-way classification. In the following we assume, that the treatment effects, say,  $u_i$  are random. That is,  $\mathbf{u} \sim N(\mathbf{A}\boldsymbol{\nu}, \sigma_b^2 \mathbf{I}_b)$ , for some design matrix  $\mathbf{A}$  and parameter vector  $\boldsymbol{\nu}$ . It can be shown that

$$\tilde{\mathbf{u}} = \frac{b\sigma_b^2}{b\sigma_b^2 + \sigma^2}(\bar{\mathbf{y}} - \mathbf{1}\bar{y}_{..}) + \frac{\sigma^2}{b\sigma_b^2 + \sigma^2}\mathbf{A}\boldsymbol{\nu} \quad (2.12)$$

where  $\bar{\mathbf{y}}$  is the vector of treatment means and  $\bar{y}_{..}$  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  $\mathbf{A}\boldsymbol{\nu}$ . If  $\boldsymbol{\nu} = \mathbf{0}$ , the BLUP in (2.12) becomes

$$\tilde{\mathbf{u}} = \frac{b\sigma_b^2}{b\sigma_b^2 + \sigma^2}(\bar{\mathbf{y}} - \mathbf{1}\bar{y}_{..}) \quad (2.13)$$

and the BLUP is a so-called shrinkage estimate. As  $\sigma_b^2$  becomes large relative to  $\sigma^2$ , the BLUP tends to the fixed effect solution, while for small  $\sigma_b^2$  relative to  $\sigma^2$  the BLUP tends towards zero, the assumed initial mean. Thus (2.13) 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  $\mathbf{X}$  can be found by summing the columns of the  $\mathbf{Z}$  matrix. This linear dependence of the matrices translates to dependence of the BLUPs and hence constraints. This aspect occurs whenever the column space of  $\mathbf{X}$  is contained in the column space of  $\mathbf{Z}$ . The dependence is slightly more complex with correlated random effects.

## 2.4 Combining variance models

The combination of variance models within G structures and R structures and between G structures and R structures is a difficult and important concept. The underlying principle is that each  $\mathbf{R}_i$  and  $\mathbf{G}_i$  variance model can only have a single overall scaling variance parameter associated with it. If there is more than one scaling variance parameter for any  $\mathbf{R}_i$  or  $\mathbf{G}_i$  then this results in the variance model being overspecified, or *nonidentifiable*. Some variance models are presented in Table 2.1 to illustrate this principle.

All of the 9 forms of model in Table 2.1 can be specified within `asreml()`. However, only models of forms 4 and 5 are recommended. Models 1-3 have too few variance parameters and are likely to cause serious estimation problems. For model 6, where the scale parameter  $\theta$  has been fitted (univariate single site analysis), it becomes the scale for  $\mathbf{G}$ . This parameterisation is bizarre and is not recommended. Models 7-9 have too many variance parameters and `asreml()` will arbitrarily fix one of the variance parameters leading to possible confusion for the user. If you fix the variance parameter to a particular value then it does not count for the purposes of applying the principle. That is, models 7-9 can be made identifiable by fixing all but one of the nonidentifiable scaling parameters in each of  $\mathbf{G}$  and  $\mathbf{R}$  to a particular value.

Table 2.1: Combination of G and R structures

model	$G_1$	$G_2$	$R_1$	$R_2$	$\theta$	comment
1.	*	*	$C$	$C$	n	invalid, no scale and $\mathbf{R}$ is a correlation model
2.	$C$	$C$	$C$	$C$	y	invalid, same scale for $\mathbf{R}$ and $\mathbf{G}$
3.	$C$	$C$	$V$	$C$	n	invalid, no scaling parameter for $\mathbf{G}$
4.	$V$	$C$	$C$	$C$	y	valid
5.	$V$	$C$	$V$	$C$	n	valid
6.	$C$	$C$	$V$	$C$	y	valid, but not recommended
7.	$V$	$V$	*	*	*	nonidentifiable, 2 scaling parameters for $\mathbf{G}$
8.	$V$	$C$	$V$	$C$	y	nonidentifiable, scale for $\mathbf{R}$ and overall scale
9.	*	*	$V$	$V$	*	nonidentifiable, 2 scaling parameters for $\mathbf{R}$

\* indicates any valid entry

Note that  $G_1$  and  $G_2$  are interchangeable in this table, as are  $R_1$  and  $R_2$

## 2.5 Inference for random effects

### 2.5.1 Tests of hypotheses

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()` 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()` this requires not only the same fixed effects model, but also the same parameterisation, as the log determinant of the matrix  $\mathbf{X}'\mathbf{X}$  is not included in the REML log-likelihood.

If  $\ell_{R2}$  is the REML log-likelihood of the more general model and  $\ell_{R1}$  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

$$D = 2 \log(\ell_{R2}/\ell_{R1}) = 2 [\log(\ell_{R2}) - \log(\ell_{R1})] \quad (2.14)$$

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  $\chi^2$  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  $0.5(1 - \Pr(\chi^2 \leq d))$ , 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  $\chi^2_1$  variate. 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  $\chi^2$  variates from 0 to  $k$  degrees of freedom. See Self and Liang [1987] for details.

Test 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

$$\begin{aligned} \text{AIC} &= -2\ell_{Ri} + 2t_i \\ \text{BIC} &= -2\ell_{Ri} + t_i \log \nu \end{aligned} \quad (2.15)$$

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.

### 2.5.2 Diagnostics

In this section we will briefly review some of the diagnostics that have been implemented in `asreml()` for examining the adequacy of the assumed variance matrix for either  $R$  or  $G$  structures, or for examining the distributional assumptions regarding  $e$  or  $u$ . Firstly we note that the BLUP of the residual vector is given by

$$\begin{aligned} \tilde{e} &= \mathbf{y} - \mathbf{W}\tilde{\boldsymbol{\beta}} \\ &= \mathbf{R}\mathbf{P}\mathbf{y} \end{aligned} \quad (2.16)$$

It follows that

$$\begin{aligned} \text{E}(\tilde{e}) &= \mathbf{0} \\ \text{var}(\tilde{e}) &= \mathbf{R} - \mathbf{W}\mathbf{C}^{-1}\mathbf{W}' \end{aligned}$$

**Hat matrix** The matrix  $\mathbf{W}\mathbf{C}^{-1}\mathbf{W}'$  is the so-called *extended hat* matrix. It is the linear mixed effects model analogue of  $\mathbf{X}(\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'$  for ordinary linear models. The diagonal elements are returned in the hat component of the `asreml` object .

**Outliers** The `aom` argument invokes a partial implementation of research by Alison Smith, Ari Verbyla and Brian Cullis. If `aom = TRUE`, `asreml()` returns

- $\mathbf{G}^{-1}\mathbf{u}$  and  $\mathbf{G}^{-1}\mathbf{u}/\text{diag}(\sqrt{\mathbf{G}^{-1} - \mathbf{G}^{-1}\mathbf{C}^{zz}\mathbf{G}^{-1}})$  as a two column matrix, and
- $\mathbf{R}^{-1}\mathbf{e}$  and  $\mathbf{R}^{-1}\mathbf{e}/\text{diag}(\sqrt{\mathbf{R}^{-1} - \mathbf{R}^{-1}\mathbf{W}\mathbf{C}^{-1}\mathbf{W}'\mathbf{R}^{-1}})$  as a two column matrix

in components named G and R, respectively, in an aom component of the fitted object.

**Variogram** 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,  $\mathbf{s}_i, i = 1, \dots, n$ , then the ordinates of the sample variogram are given by

$$v_{ij} = \frac{1}{2} [e_i(\mathbf{s}_i) - e_j(\mathbf{s}_j)], \quad i, j = 1, \dots, n; \quad i \neq j$$

The sample variogram is 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. If the data arise from a regular array there will be many  $v_{ij}$  with the same absolute displacements, in which case `plot.asreml()` displays the vector  $(l_{ij1}, l_{ij2}, \bar{v}_{ij})$  as a perspective plot.

**variogram()** 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.

## 2.6 Inference for fixed effects

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()`. 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  $H_0 : \mathbf{L}\boldsymbol{\tau} = \mathbf{l}$  for given  $\mathbf{L}, r \times p$ , and  $\mathbf{l}, r \times 1$ , is given by

$$\mathcal{W} = (\mathbf{L}\hat{\boldsymbol{\tau}} - \mathbf{l})' \{ \mathbf{L}(\mathbf{X}'\mathbf{H}^{-1}\mathbf{X})^{-1}\mathbf{L}' \}^{-1} (\mathbf{L}\hat{\boldsymbol{\tau}} - \mathbf{l}) \quad (2.17)$$

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()`) of settings.

In the following we describe the facilities currently available in `asreml()` 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.

### 2.6.1 Incremental and Conditional Wald Statistics

The basic tool for inference is the Wald statistic defined in equation 14.1. However, there are several ways  $L$  can be defined to construct a test for a particular model term, two of which are available in `asreml()`. 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

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

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

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

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  $R(B|1, A)$  represents the difference between the reduction in sums of squares between the *maximal* model

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

and

$$y \sim 1 + A$$

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

$$R(A|1, B) = R(1, A, B) - R(1, B)$$

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()`, the so-called *conditional* Wald statistic. This method is invoked by specifying `ssType = conditional` in `wald.asreml()`. `asreml()` 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

$$y \sim 1 + \text{REGION} + \text{REGION.SITE}$$

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

$$y \sim 1 + \text{REGION} + \text{SITE}$$

with SITE identified across REGION rather than within REGION. Then the nested structure is hidden but `asreml()` 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  $\nu_{1i}$ ) 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"`.

## 2.7 Kenward and Roger Adjustments

In moderately sized analyses, `asreml()` can also calculate the denominator degrees of freedom (`DenDF`, denoted by  $\nu_{2i}$ , [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  $\hat{\tau}$ . They argued that it is useful to consider an improved estimator of the variance matrix of  $\hat{\tau}$  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  $\hat{\tau}$  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.

## 2.8 Approximate stratum variances

[svc method](#) 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  $\sigma_H^2 = 1$ ) where  $\mathbf{G} = \oplus_{j=1}^q \gamma_j \mathbf{I}_{b_j}$ , it is often possible to consider a natural ordering of the variance component parameters including  $\sigma^2$ . Based on an idea due to Thompson [1980] `asreml()` computes approximate stratum degrees of freedom and stratum variances by a modified Cholesky diagonalisation of the expected (or average) information matrix. That is, if  $\mathbf{F}$  is the average information matrix for  $\boldsymbol{\sigma}$ , let  $\mathbf{U}$  be an upper triangular matrix such that  $\mathbf{F} = \mathbf{U}'\mathbf{U}$ . Further we define

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

where  $\mathbf{D}_c$  is a diagonal matrix whose elements are given by the inverse elements of the last column of  $\mathbf{U}$  ie  $d_{cii} = 1/u_{ir}, i = 1, \dots, r$ . The matrix  $\mathbf{U}_c$  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  $\sigma^2$  being the last element, then we can define the vector of so called *pseudo* stratum variance components by

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

Thence

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

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.

# Fitting the mixed model

## 3

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 8 gives a lengthy set of additional worked examples.

### 3.1 The data frame

Data for analysis using `asreml()` 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,Rep,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
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  $V_1 \dots V_n$  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 `Rep` 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. Thus 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`, `Rep`, `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$Rep <- factor(nin89$Rep)
> nin89$Row <- factor(nin89$Row); nin89$Column <- factor(nin89$Column)
```

## 3.2 Introducing the `asreml()` function call

The 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 = ~ Rep,
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 `rcov` (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.

**Fixed terms** The fixed terms in the model are specified as a formula with the response on the left of a `~` 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. Thus, if the intercept was the only fixed term in the model then the `fixed` argument would be

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

**Random terms** 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 `~` operator. In this example `Rep` is a random term so the `random` argument is

```
nin89.asr <- asreml(..., random = ~ Rep, ...)
```

**Error terms** The residual or error component of the model is specified in a formula object through the `rcov` argument. The default is a simple error term 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(..., rcov = ~ 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()` 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 are given in Section 7.3.

### 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),

`summary()` The `summary.asreml()` function returns a list with a range of components:

```
> names(summary(nin89.asr))
[1] "call"          "distribution"  "link"         "loglik"
[5] "nedf"          "sigma"        "deviance"     "heterogeneity"
[9] "varcomp"      "coef.fixed"   "coef.random"  "coef.sparse"
[13] "residuals"
```

**Components** The variance components are returned in

```
> summary(nin89.asr)$varcomp
      gamma component std.error  z.ratio constraint
Rep      0.1993231  9.882913  8.792685  1.123993  Positive
R!variance 1.0000000 49.582378  5.458841  9.082950  Positive
```

**Coefficients** 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.fixed
              solution std error      z ratio
Variety_ARAPAHOE  0.0000      NA           NA
Variety_BRULE    -3.3625  4.979087 -0.675324649
Variety_BUCKSKIN -3.8750  4.979087 -0.778255171
...
Variety_TAM200   -8.2000  4.979087 -1.646888363
Variety_VONA     -5.8375  4.979087 -1.172403758
(Intercept)     29.4375  3.855601  7.634996452
```

### 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 `rcov` 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 `rcov` formula. In the case, for example, where

```
rcov = ~ ar1(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 Chapter 7 and can be obtained from the help system within R:

```
> ?areml
or
> help(asreml)
```

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

On Windows systems, the `asreml.chm` help file stored in the ASReml-R installation directory and on all systems, this manual (`asreml.pdf`) is available in the ASReml-R installation tree.

### 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 `~` in the list of primary fixed terms, for example, use

```
> 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 = ...)
```

**Special functions** 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 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
<b>reserved names</b>		
<code>mv</code>	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 <code>na.method.Y</code> argument. If <code>na.method.Y = "omit"</code> , records containing missing values in the response are deleted. If <code>na.method.Y = "include"</code> , missing values are estimated and a factor labelled <code>mv</code> included in the model frame. If a variate labelled <code>mv</code> already exists in the data frame it will be overwritten. For a multivariate analysis, missing values must currently be included	f
<code>trait</code>	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 <code>trait</code> already exists in the data frame it will be overwritten.	f, r
<code>units</code>	a factor with a level for each experimental unit; allows a second error term to be explicitly fitted.	r
<b>model functions</b>		

## Summary of reserved names special functions

term	purpose	usage
at(f,l)	condition on level $l = 1, \dots, k$ 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 <code>at(site,3):row</code> . Note that if $l$ is numeric, then the level of $f$ is chosen as the $l$ th in factor (sorted) order. Note also that when used with spline terms, such as <code>at(f,2):spl(x)</code> then the knot points are derived from <b>all</b> of factor $f$ , <b>not</b> just level 2.	f, r
dev(x)	Forms a factor with a level for each unique value of $x$ .	r
grp(obj)	Groups contiguous columns of <b>data</b> to be treated as a single factor named "obj". The columns of <b>data</b> are identified by a character or numeric vector component <b>obj</b> of the <b>group</b> argument to <code>asreml.control()</code> .	r
lin(f)	treats the named factor as a variate. The function is defined for $f$ being a simple factor, trait and units. The <code>lin(f)</code> function does not center or scale the variable.	f, r
link(a,b)	ensures that the structures for terms <b>a</b> and <b>b</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 <b>obj</b> as a set of covariates to be fitted as a single term in a similar way to <b>grp</b> . The name <b>obj</b> must also appear as a component of the <b>mbf</b> argument to <code>asreml.control()</code> where the data frame holding the covariates is identified along with a key field for merging records with those in <b>data</b> .	r
pol(x,t)	forms $t$ orthogonal polynomials from the values in $x$ ; the mean is excluded if $t$ is negative. For example, <code>pol(time,2)</code> 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. <code>pol()</code> could be interacted with a design factor to fit random regression models.	f, r
spl(x, k, points)	Random component of a cubic spline for covariate $x$ . <code>spl(x)</code> , <code>dev(x)</code> and possibly <code>lin(x)</code> 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, <code>spl(x)</code> , and the fixed covariate ( $x$ ) in the linear model. Knot points are placed at the design points if <code>length(unique(x)) &lt; k</code> otherwise there are $k$ equally spaced knot points over the range of $x$ . The default for $k$ is 50. Alternatively, <b>points</b> may contain a vector of user specified knot points. Both <b>k</b> and <b>points</b> may be omitted and defaults set in <code>asreml.control()</code> .	r

### 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()` 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 defaults to a scaled identity  $\gamma \mathbf{I}_n$  or  $\sigma^2 \mathbf{I}_n$  where  $\gamma$  is a variance ratio. See page 15 under 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 R algebraically, see Chapter 4) and the variance matrix for the other random terms in the model (referred to as G structures and denoted  $\mathbf{G}$  algebraically). 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 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 = ~ Rep, data = nin89)
```

a single variance component ratio is estimated for the random Rep 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()`.

### Replacing elements in an internal object

For example, to set a different initial value for the `Rep` component, the call

```
> nin89.sv <- asreml(fixed = yield ~ Variety, random = ~ Rep,
  na.method.X = "include", data = nin89, start.values = TRUE)
```

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

```
> iv <- nin89.sv$gammas.table
> iv
      Gamma Value Constraint
1      Rep    0.1          P
2 R!variance 1.0          P
```

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

```
> nin89.asr <- asreml(fixed = yield ~ Variety, random = ~ Rep, na.method.X = "include",
  data = nin89, G.param = iv)
```

### 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 `gammas.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 = ~ Rep,
  na.method.X = "include", data = nin89, start.values = "filename" )
```

with the revised values included in the analysis by:

```
> nin89.asr <- asreml(fixed = yield ~ Variety, random = ~ Rep,
  na.method.X = "include", data = nin89, G.param = "filename" )
```

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

### 3.7.2 Specifying variance structures

Beyond IID

The default variance model for a term in the random model is a scaled identity ( $\gamma\mathbf{I}_n$  or  $\sigma^2\mathbf{I}_n$ ), 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 (`rcov`) 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

```
rcov = ~ ar1(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() function

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):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)`. When used in an `rcov` formula, `at()` specifies a variance model for `e` as a direct sum of `l` variance matrices, one for each level of the conditioning factor.

## 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 = asreml.gaussian(dispersion=1.0),...)
```

## 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.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.method.Y = "include"`) and 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.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  $\eta$  to the mean ( $\mu$ ) 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	$\eta = \mu/(1 - \mu)$		D		
probit	$\eta = \Phi^{-1}(\mu)$		*		
cloglog	$\eta = \log(-\log(1 - \mu))$		*		

where  $\mu$  is the mean on the data scale,  $\boldsymbol{\eta} = \mathbf{X}\boldsymbol{\tau}$  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 form of cross-validatory assessment for the specific dataset 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  $y_1, \dots, y_k$  in the data frame, a multivariate analysis would be specified with the call

```
> asreml(fixed = cbind(y1, ..., yk) ~ 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 `as.multivariate = 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 `rcov` 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.

#### 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` dataset 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,
+ rcov = units:us(trait,init=rep(0,15)), maxiter=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
```

	gamma	component	std.error	z.ratio	constraint
R!variance	1.00000	1.00000	NA	NA	Fixed
R!trait_wt0:wt0	21.57560	21.57560	6.228322	3.464110	Unconstrained
R!trait_wt1:wt0	33.01964	33.01964	10.354376	3.188955	Unconstrained
R!trait_wt1:wt1	68.72738	68.72738	19.839816	3.464114	Unconstrained
R!trait_wt2:wt0	31.58214	31.58214	11.258510	2.805180	Unconstrained
R!trait_wt2:wt1	69.06071	69.06071	21.681866	3.185183	Unconstrained
R!trait_wt2:wt2	94.76905	94.76905	27.357257	3.464128	Unconstrained
R!trait_wt3:wt0	29.37619	29.37619	14.112774	2.081532	Unconstrained
R!trait_wt3:wt1	64.54345	64.54345	26.334015	2.450954	Unconstrained
R!trait_wt3:wt2	116.35595	116.35595	35.791126	3.250972	Unconstrained
R!trait_wt3:wt3	181.55655	181.55655	52.410398	3.464132	Unconstrained
R!trait_wt4:wt0	24.66071	24.66071	16.328866	1.510253	Unconstrained
R!trait_wt4:wt1	56.72024	56.72024	30.044386	1.887881	Unconstrained
R!trait_wt4:wt2	122.88690	122.88690	41.098144	2.990084	Unconstrained
R!trait_wt4:wt3	207.21310	207.21310	61.801813	3.352864	Unconstrained
R!trait_wt4:wt4	268.40952	268.40952	77.482498	3.464131	Unconstrained

### 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(0.2,0.2,2.0)):Tag,
rcov = ~ units:us(trait,init=c(0.2,0.2,0.4)), data = orange)
```

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

```
> wether.asr$monitor[,c(1,'final','constraint')]

```

	1	final	constraint
loglik	-886.5213	-723.4616740	<NA>
S2	1.0000	1.0000000	<NA>
df	2964.0000	2964.0000000	<NA>
trait:Team!trait_gfw:gfw	0.4000	0.3744933	Unconstrained
trait:Team!trait_fdiam:gfw	0.3000	0.3887395	Unconstrained
trait:Team!trait_fdiam:fdiam	1.3000	1.3653342	Unconstrained
trait:Tag!trait_gfw:gfw	0.2000	0.2571589	Unconstrained
trait:Tag!trait_fdiam:gfw	0.2000	0.2195574	Unconstrained
trait:Tag!trait_fdiam:fdiam	2.0000	1.9208175	Unconstrained
R!variance	1.0000	1.0000000	Fixed
R!trait_gfw:gfw	0.2000	0.1983514	Unconstrained
R!trait_fdiam:gfw	0.2000	0.1288901	Unconstrained
R!trait_fdiam:fdiam	0.4000	0.4406009	Unconstrained

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')
```

```

          Df  denDF F_inc F_con Margin Pr
trait      2   33.0  5762  5762      A  0
trait:Year  4 1162.2  1095  1095      B  0

```

### 3.13.2 Specifying multivariate variance structures

A more sophisticated default error structure is required for multivariate analysis in `asreml()`. 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

$$I_n \otimes \Sigma$$

where  $\Sigma^{(n_t \times n_t)}$  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 `rcov` this model is therefore `rcov ~ 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 `rcov` 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:

```
> wolfinger.asr <- asreml(fixed = cbind(wt0,wt1,wt2,wt3,wt4) ~ trait * Treatment,
+ rcov = 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  $\chi_k^2$  where  $k$  is the number of nonsingular effects in the term.

#### *Conditional F-statistics and denominator degrees of freedom*

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 2.6 for more background on the contents of this table.

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()` will only attempt the calculation if specifically requested as it requires further iterations of the model (using `update.asreml()`).

# Specifying variance structures

## 4

This chapter introduces variance model specification in `asreml()`, a complex aspect of the modelling process. The key concepts are:

- The mixed linear model

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

has a residual term

$$\mathbf{e} \sim N(\mathbf{0}, \theta\mathbf{R})$$

and random effects

$$\mathbf{u} \sim N(\mathbf{0}, \theta\mathbf{G})$$

where in the most complex forms

$$\mathbf{R} = \oplus_i \mathbf{R}_i$$

$$\mathbf{G} = \oplus_j \mathbf{G}_j$$

and each

$$\mathbf{R}_i = \mathbf{R}_i(\boldsymbol{\phi}_i)$$

$$\mathbf{G}_j = \mathbf{G}_j(\boldsymbol{\delta}_j)$$

where  $\boldsymbol{\phi}_i$  and  $\boldsymbol{\delta}_j$  parameterise the respective variance models.

- We use the terms R structure and G structure to refer to the matrices  $\mathbf{R}_i$  and  $\mathbf{G}_j$  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`),

- Variances are sometimes estimated as variance component ratios (relative to the overall scale parameter,  $\theta$ ).

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

**Special functions** Variance models are specified with special model functions in the `random` and `rcov` 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()` 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, seven variance structures of increasing complexity are considered for the NIN field trial data (see Section 1.3.1). This is to give a general feel for variance modelling in `asreml()` from a practical perspective and some idea of the types of models that are possible (Table B.1).

This section illustrates:

- changes to  $\mathbf{u}$  and  $\mathbf{e}$  and the assumptions regarding the variance these terms,
- the impact this has on the `random` formula for specifying the G structures for  $\mathbf{u}$  and the `rcov` formulae for specifying the R structure(s) for the residuals in  $\mathbf{e}$ .

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

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

The only random term in this analysis is the residual term where we have assumed  $\mathbf{e} \sim N(\mathbf{0}, \theta \mathbf{I}_{224})$ . The model therefore involves just one R structure and no G terms. In `asreml()`

- the scaled variance structure ( $\mathbf{R} = \theta \mathbf{I}_{224}$ ) 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 `rcov` argument,

The `asreml()` call above is therefore equivalent to

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

or more specifically

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

where `idv()` is the special model function in `asreml()` that identifies the variance model. The expression `idv(units)` explicitly sets the variance matrix for  $\mathbf{e}$  to a scaled identity.



The error term is *always* present in the model and does not need to be explicitly declared when it has the default structure.

### Model 1a: RCB analysis - blocks random

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

This specifies  $\mathbf{u}$  as a vector of `Replicate` effects where  $\text{var}(\mathbf{u}) = \gamma_r \mathbf{I}_4$ ,  $\gamma_r = \sigma_r^2 / \theta$  and assumes that  $\text{var}(\mathbf{e}) = \theta \mathbf{I}_{24}$ .

Note that to obtain the REML estimate of the variance component for `Replicate` ( $\sigma_r^2$ ) we compute

```
> rcb.asr$gammas["Replicate"]*rcb.asr$sigma
```

This is done automatically by the summary method (`summary.asreml()`) where both variance components and variance ratios are returned.

In `asreml()`, the IID variance structure is the default for the extra random terms in the model and does not need to be formally specified in the `random` formula.



All random terms other than residual error must appear in the `random` formula (see Section 3.7).

### Model 1b: RCB analysis with G and R structures

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

This model is equivalent to **1a** and introduces the use of variance model functions in the `random` and `rcov` formulae to explicitly specify the G and R structures. In practice it is usually not necessary to specify the default variance models unless setting initial values or boundary constraints.

Note that when specifying G structures, the user must ensure that one scale parameter is present; `asreml()` does not automatically include it. *All but one* of the models specified in a G structure *must* be correlation models; the other must be a variance model. If the variance matrix of a term contains several component matrices the the problem of *identifiability* arises. For example,



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

produces a variance matrix of the form  $\gamma_a \gamma_b \mathbf{I}_{a,b}$  for A:B where the  $\gamma_A$ ,  $\gamma_B$  parameters are not identifiable.

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

```
> sp.asr <- asreml(yield ~ Variety, rcov = ~ 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  $\text{var}(\mathbf{e}) = \theta(\mathbf{I}_{11} \otimes \boldsymbol{\Sigma}_r)$ . The R structure is the direct product of two matrices; an 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  $j$ . The variance model for `Column` is identity (`id()`) but does not need to be formally specified as this is the default. Note that

**Data order**

- the direct product structure is implied by the ":" operator. The order in which factors appear in the `rcov` 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()` does not reorder the observations; if the data frame is not in the order specified by `rcov` 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()` automatically includes missing values in the sparse component with a factor named `mv` (see Section 3.10).

- unlike G structures, `asreml()` automatically includes and estimates  $\theta$ . In this example the variance models specified for `Row` (`ar1()`) and `Column` (default `id()`) are correlation models. If the R structure is a variance matrix then the parameter  $\theta$  must be constrained to 1.0 (using the `dispersion` argument to the appropriate family function, such as `asreml.gaussian()`). `asreml()` attempts to detect these situations but it is wise to explicitly constrain  $\theta$ . Specifically, the call

```
> sp.asr <- asreml(yield ~ Variety, rcov = ~ idv(Column):ar1(Row), data = nin89,
+ family = asreml.gaussian(dispersion = 1.0))
achieves this.
```

**Model 2a: two-dimensional spatial model**

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

This extends model 2 by specifying a first order autoregressive correlation model of order 11 for columns (`ar1()`). The R structure in this case is therefore the direct product of two autoregressive correlation matrices that is,  $\text{var}(\mathbf{e}) = \theta(\boldsymbol{\Sigma}_c \otimes \boldsymbol{\Sigma}_r)$ .

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

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

This model includes a factor with  $n = 224$  levels in  $\mathbf{u}$ . Since  $\mathbf{Z} = \mathbf{I}$ ,  $\text{var}(\mathbf{y}) = \theta(\gamma\mathbf{I}_{224} + \boldsymbol{\Sigma}_c \otimes \boldsymbol{\Sigma}_r)$ . The quantity  $\theta\gamma$  is the so-called measurement error variance or nugget variance in geostatistics. `units` is a reserved name that `asreml()` constructs internally as `seq(1,nrow(data))`. Again, the default `idv()` variance model is used for `units`.

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

```
> sp.asr <- asreml(yield ~ Variety, random = ~ ar1v(Column):ar1(Row), 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 **1b**,

Combining  
variance models

- when the G structure term involves more than one model, all but one of the models must be a correlation model (see Section 4.5). In this example (`ar1v()`) is the variance model.
- an initial value for the scale parameter in this model must be supplied; the `asreml()` generated default (0.1) is used here.

Modelling `Column:Row` as a G structure is a useful approach to handling incomplete arrays.

## 4.2 Types of variance models

There are three types of variance model that are used in fitting R and G structures in `asreml()`, 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 `rcov` by  $\boldsymbol{\Sigma}$ .

### 4.2.1 Correlation models

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

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

The simplest correlation model in `asreml()` is the `id()` model, where  $\boldsymbol{\Sigma} = \mathbf{I}_\omega$

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

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

### 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,  $\sigma^2$  say. That is,

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

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

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

### 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,

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

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

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

#### 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 require these matrices to be either negative definite or positive definite. They must not be singular. Negative definite matrices will have negative elements on the diagonal of the matrix and/or its inverse. The exception is the `fa` model which has been specifically designed to fit singular matrices [Thompson et al., 2003].

### 4.3 Variance model functions

`asreml()` has a wide range of variance models which can be used to specify the variance matrix of terms in the random and rcov 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.

<b>model</b>	<b>form:</b>	<b>number of parameters</b>		
<i>f</i>		<i>f()</i>	<i>f<math>\mathbf{v}</math>()</i>	<i>f<math>\mathbf{h}</math>()</i>
id		0	1	<i>n</i>

### *Details*

`asreml()` 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 `rcov` 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 ( <i>f</i> )	form:	number of parameters		
		<i>f</i> ()	<i>fv</i> ()	<i>fh</i> ()
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	$2 + n$

### Details

#### 4.3.3 Metric based models in $\mathcal{R}$ or $\mathcal{R}^2$

```
exp(x, init=NA, dist=NA)
gau(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 `pwrpoints` argument to `asreml.control()`.

#### 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.

<b>model</b> ( <i>f</i> )	<b>form:</b>	<b>number of parameters</b>		
		<i>f</i> ()	<i>f</i> <b>v</b> ()	<i>f</i> <b>h</b> ()
exp		1	2	$1 + n$
gau		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:  $\phi = \text{NA}$ .

`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 `pwrpoints` list argument to `asreml.control()` (Section 7.2)

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, otherwise an error is generated.

#### The Matérn class

`asreml()` uses an extended Matérn class which accomodates 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

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

where  $\mathbf{h} = (h_x, h_y)^T$  is the spatial separation vector,  $(\delta, \alpha)$  governs geometric anisotropy,  $(\lambda)$  specifies the choice of metric and  $(\phi, \nu)$  are the parameters of the Matérn correlation function. The function is

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

where  $\phi > 0$  is a range parameter,  $\nu > 0$  is a smoothness parameter,  $\Gamma(\cdot)$  is the gamma function,  $K_\nu(\cdot)$  is the modified Bessel function of the third kind of order  $\nu$  (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  $\nu$ , the range parameter  $\phi$  affects the rate of decay of  $\rho(\cdot)$  with increasing  $d$ . The parameter  $\nu > 0$  controls the analytic smoothness of the underlying process  $\mathbf{u}_s$ , the process being  $[\nu] - 1$  times mean-square differentiable, where  $[\nu]$  is the smallest integer greater than or equal to  $\nu$  (Stein, 1999, page 31). Larger  $\nu$  correspond to smoother processes. `asreml()` uses numerical derivatives for  $\nu$  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  $\nu \rightarrow \infty$  then  $\rho_M(\cdot)$  tends to the gaussian correlation function.

The metric parameter  $\lambda$  is not estimated by `asreml()`; 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  $\text{AR1} \times \text{AR1}$  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  $\nu$  is often set a-priori [Kammann and Wand, 2003, Diggle et al., 2003]. However Stein [1999] and Haskard et al. [2005] demonstrate that  $\nu$  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 ( $\phi$ ,  $\nu = 0.5$ ,  $\delta = 1$ ,  $\alpha = 0$ ,  $\lambda = 2$ ). 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  $\phi$  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  $\nu$  some value other than the default and estimate  $\phi$ , 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  $\phi$  and  $\nu$  are highly correlated so it may be better to manually cover a grid of  $\nu$  values.

We note that there is non-uniqueness in the anisotropy parameters of this metric  $d(\cdot)$  since inverting  $\delta$  and adding  $\frac{\pi}{2}$  to  $\alpha$  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  $\alpha$  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)
fa(obj, k=1, init=NA)
```

**Description**

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	number of parameters			
( <i>f</i> )	form:	<i>f</i> ( )	<i>f</i> <b>v</b> ( )	<i>f</i> <b>h</b> ( )
<code>cor</code>		1	2	1 + <i>n</i>
<code>corb</code>		<i>k</i> - 1	<i>k</i>	2 <i>k</i> - 1
<code>corg</code>		$n(n-1)/2$	$1 + n(n-1)/2$	$n + n(n-1)/2$
<code>diag</code>		<i>n</i>		
<code>us</code>		$n(n+1)/2$		
<code>chol</code> <sup>1</sup>		$n(n+1)/2$		
<code>cholc</code> <sup>1</sup>		$n(n+1)/2$		
<code>ante</code> <sup>1</sup>		$n(n+1)/2$		
<code>fa</code>		<i>kn</i> + <i>n</i>		

<sup>1</sup> `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**

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

$$\Sigma = LDL'$$

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

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

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

**cholc** In the `cholc(,k)` factorization  $L$  has columns  $\mathbf{l}_i = (l_{1i}, \dots, l_{\omega i})'$  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  $\Sigma = LDL'$  where

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

This form is similar to a Factor Analytic model. In `asreml()` 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

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

the initial values are given as

`c(σ11, σ21, σ22, ..., σ44)`

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

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

where  $U^{\omega \times \omega}$  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  $D = \text{diag}(d_1, \dots, d_\omega)$ .

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

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

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

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

In `asreml()` the parameters for `ante` are given as `for us`, `chol` and `cholc` as the lower triangle row-wise of an unstructured matrix.

**factor analytic** In a factor analytic model of order  $k$  (`fa(,k)`), the variance matrix  $\Sigma^{\omega \times \omega}$  is modelled

as

$$\Sigma = \Gamma\Gamma' + \Psi$$

where  $\Gamma$  ( $\omega \times k$ ) is a matrix of loadings and  $\Psi^{\omega \times \omega}$  is a diagonal matrix whose elements are referred to as specific variances.

For example, if Site is a factor with 4 levels, the component matrices for

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

are

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

where the parameters are given in the order `c(diag( $\Psi$ ),vec( $\Gamma$ ))`.

A key issue with factor analytic models is that it is possible for the REML estimates of the specific variances to be zero. The `fa()` algorithm [Thompson et al., 2003] used in `asreml()` permits some (or all) specific variances to be set to zero.

If  $k > 1$ , constraints on the elements of  $\Gamma$  are required for identifiability. These constraints are chosen to be:  $l_{12} = 0$  for  $k = 1$ ;  $l_{13} = l_{23} = 0$  for  $k = 3$ , etc. `asreml()` sets these elements to zero and their corresponding boundary constraints to "F".

### 4.3.5 Known relationship structures

`giv(obj, init=NA)`

`ped(obj, init=NA)`

#### **Required arguments**

`obj` a factor in the data frame. The name `obj` must also appear as a component in the `ginverse` list argument to `asreml.control()` to associate an inverse relationship matrix with the factor `oj`

#### **Optional arguments**

`init` a vector of length 1 giving the initial variance parameter value. This scalar can have an optional `names` attribute to set the boundary constraint. In this case, the name may be one of "P", "U" or "F" for positive, unconstrained or fixed, respectively.

#### **Details**

The `giv()` procedure associates a known inverse matrix with the factor `obj`; the number of rows in the inverse matrix must be `length(levels(obj))` and the order is assumed correct. The `ped()` function associates an inverse relationship matrix typically derived from a pedigree with the factor `obj`. There may be more rows in the inverse matrix than levels of `obj`. More than one inverse matrix may be used in an analysis so the `ginverse` argument to `asreml()` must be used in conjunction with `giv` or `ped` to associate particular inverse matrices with factors in the model.

### 4.3.6 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 random regressions case we would generally wish to estimate a covariance between intercept and slope. The syntax for a longitudinal analysis of animal liveweight over time, say, is

```
> asreml(..., random = ~ str(~ Animal+Animal:Time, ~ us(2):id(25)), ...)
```

assuming the factor **Animal** has 25 levels. See Section 8.9 for an example of its use in fitting random coefficient models.

## 4.4 Default initial values for variance parameters

The default initial values are 0.1 for both variance ratios and correlations, and  $0.1 \cdot v$  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

As discussed in Section 2.4, 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 43) the `ar1v()` variance model for Column is specified first in the interaction term.
- `asreml()` automatically includes and estimates a error variance parameter for each section of an R structure. The variance structures defined by the user should therefore normally be correlation matrices. A variance model can be specified but the dispersion parameter in the `family` argument must then be set to 1.0 to fix the error variance at 1 and prevent `asreml()` trying to estimate two confounded parameters (error variance and the parameter corresponding to the variance model specified, see **Model 2** on page 42).
- `asreml()` does not have an implicit scale parameter when G structures are defined in the `random` model formula. For this reason one, and only one, of the models in the G structure term must be a variance function; an initial value must be supplied for the associated scale parameter, this is discussed under Initial values and constraints for variance parameters on page 30.
- 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.

See Sections 2.1  
and B

## 4.6 Constraining variance parameters

Equality and more general relationships among variance parameters are specified in `asreml()` with a simple linear model. Let  $\kappa$  be the  $n_\kappa$  vector of unconstrained variance parameters and  $T$  be a  $n_\kappa \times n_c$  matrix imposing the linear constraints  $T'\kappa = s$ . This is equivalent to

$$\kappa = M\theta + Es$$

where  $\theta$  is the  $n_c$  vector of constrained parameters. Constraints are specified in `asreml()` by the matrix  $M$  which must have a `dimnames` attribute with the names of  $\kappa$  as its row names.



Note that in `asreml()`  $n_\kappa$  need only encompass the subset of variance parameters among which constraints will be applied, rather than the entire set.

The function `asreml.constraints()` generates the matrix  $M$  from a linear model formula and a data frame in which to

1. resolve the terms named in that formula, and
2. extract the variance component names as the `dimnames` attribute of  $M$ .

A default data frame `gammas.table` to work with is generated by setting the `start.values` argument to `asreml()`. This data frame contains a factor, `Gammas`, the levels of which are the names of the variance parameters. Other factors or variates created in this data frame and named in the formula argument to `asreml.constraints()` are used to generate  $M$ .

**an example** By way of example, consider a model containing a first order interaction term ( $\mathbf{A}:\mathbf{B}$ , say) where the outer factor ( $\mathbf{A}$ ) is of order 7 and we wish to model it with an unstructured variance matrix with some parameters constrained. Suppose the required pattern of constraints among the variance parameters is:

```

v11
v21 v22
v31 v32 v33
v31 v32 0 v33
v31 v32 0 0 v33
v31 v32 0 0 0 v33
v31 v32 0 0 0 0 v33

```

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):B, start.values = "gammas.txt", ...)
```

generates

1. a text file "`gammas.txt`" containing the component names, their initial values and boundary constraints, and
2. a data frame component of `model.gam` named `gammas.table` with the same contents as the above file.

If the 28 components of interest are the 47<sup>th</sup> through 74<sup>th</sup> in the `gammas` vector, for example, the following code subsets `model.gam$gammas.table` and creates a factor in the reduced table that can be used to construct  $M$ :

```

gam <- model.gam$gammas.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,
4,5,7,7,7,6))

M <- asreml.constraints(~ fac, gammas=gam)

```

`asreml.constraints()` omits the constant by default and calls `model.frame()` to generate  $M$ . Note that any procedure could be substituted for `asreml.constraints()` provided the resulting matrix conforms.

In this example,  $M$  would look like

```
      1  2  3  4  5  6  7
v1,1  1  0  0  0  0  0  0
v2,1  0  1  0  0  0  0  0
v2,2  0  0  1  0  0  0  0
v3,1  0  0  0  1  0  0  0
v3,2  0  0  0  0  1  0  0
v3,3  0  0  0  0  0  1  0
v4,1  0  0  0  1  0  0  0
v4,2  0  0  0  0  1  0  0
v4,3  0  0  0  0  0  0  1
v4,4  0  0  0  0  0  1  0
v5,1  0  0  0  1  0  0  0
v5,2  0  0  0  0  1  0  0
v5,3  0  0  0  0  0  0  1
v5,4  0  0  0  0  0  0  1
v5,5  0  0  0  0  0  1  0
v6,1  0  0  0  1  0  0  0
v6,2  0  0  0  0  1  0  0
v6,3  0  0  0  0  0  0  1
v6,4  0  0  0  0  0  0  1
v6,5  0  0  0  0  0  0  1
v6,6  0  0  0  0  0  1  0
v7,1  0  0  0  1  0  0  0
v7,2  0  0  0  0  1  0  0
v7,3  0  0  0  0  0  0  1
v7,4  0  0  0  0  0  0  1
v7,5  0  0  0  0  0  0  1
v7,6  0  0  0  0  0  0  1
v7,7  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 editing `gammas.txt` and setting the appropriate values to zero and boundary constraint codes to F. The modified values and the matrix  $M$  are used through the `G.param` and `constraints` arguments to `asreml()`, respectively. That is

```
model.asr <- asreml(..., random = us(A):B, constraints = M, G.param = "gammas.txt", ...)
```

# Genetic analysis

## 5

### 5.1 Introduction

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  $\mathbf{A}$  matrix) can be calculated from the pedigree. It is actually the **inverse** relationship matrix that is required by `asreml()` for analysis.

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

1. the function `asreml.Ainverse()` takes a pedigree data frame and returns the  $\mathbf{A}^{-1}$  matrix in sparse form as a `giv` object (see below).
2. The matrix from step 1 (`giv` object) is included in an `asreml()` analysis using the `ginverse` argument in conjunction with the `ped()` variance model function.

For the more general situation, where the pedigree based inverse relationship matrix generated by `asreml.Ainverse()` is not appropriate, the user can include a general inverse variance matrix provided its structure adheres to one of the allowable forms for a `giv` object (see below).

There may be more than one G-inverse matrix present and each are supplied through named components of the `ginverse` argument. The `ped()` and `giv()` special functions in the random model formula associate the appropriate G-inverse with the nominated model factor.

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

### 5.2 Pedigree, G-inverse objects 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 `asreml.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)
> harvey.ped[1:20,]
      Calf  Sire Dam
1 Sire_1    0  0
2 Sire_2    0  0
3 Sire_3    0  0
4 Sire_4    0  0
5 Sire_5    0  0
6 Sire_6    0  0
7 Sire_7    0  0
8 Sire_8    0  0
9 Sire_9    0  0
10  101 Sire_1  0
11  102 Sire_1  0
12  103 Sire_1  0
13  104 Sire_1  0
14  105 Sire_1  0
15  106 Sire_1  0
16  107 Sire_1  0
17  108 Sire_1  0
18  109 Sire_2  0
19  110 Sire_2  0
20  111 Sire_2  0
```

### 5.2.2 giv objects

An inverse relationship matrix,  $\mathbf{A}^{-1}$  or a general inverse matrix from an external source can be included in the analysis as:

- a data frame of three columns containing the non-zero elements of the lower triangle of the matrix in row order. The first two columns of the data frame are the row and column indices and must be named *row* and *column*, respectively. This is the structure returned by `asreml.Ainverse()`.
- a matrix object. Only the lower triangle is used and NAs are ignored.
- the complete lower triangle in vector form stored row by row; NAs are ignored.
- in all cases every diagonal element must be present.



In all cases the `giv` 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 <- asreml.Ainverse(harvey.ped)$ginv
> attr(harvey.ainv,"rowNames")
 [1] "Sire_1" "Sire_2" "Sire_3" "Sire_4" "Sire_5" "Sire_6" "Sire_7" "Sire_8"
 [9] "Sire_9" "101"  "102"  "103"  "104"  "105"  "106"  "107"
[17] "108"   "109"  "110"  "111"  "112"  "113"  "114"  "115"
[25] "116"   "117"  "118"  "119"  "120"  "121"  "122"  "123"
[33] "124"   "125"  "126"  "127"  "128"  "129"  "130"  "131"
[41] "132"   "133"  "134"  "135"  "136"  "137"  "138"  "139"
[49] "140"   "141"  "142"  "143"  "144"  "145"  "146"  "147"
[57] "148"   "149"  "150"  "151"  "152"  "153"  "154"  "155"
[65] "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
```

### 5.2.3 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 `asreml.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)
> harveyG.ped[1:20,]
      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 Generating an A-inverse matrix with `asreml.Ainverse()`

`asreml.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 `asreml.Ainverse()` is given in Section 7.4.2.

### 5.4 Using Pedigree and G-inverse objects

Putting it all together, an analysis of *average daily gain* (`y1` in `harvey.dat`) using the pedigree `harvey.ped` can be obtained from:

```

> harvey.ainv <- asreml.Ainverse(harvey.ped)$ginv
> adg.asr <- asreml(y1 ~ Line, random = ~ ped(Calf, var=T), ginverse=list(Calf=harvey.ainv), data=harvey)

```

The variance components are given in

```

> summary(adg.asr)$varcomp

```

	gamma	component	std.error	z.ratio	constraint
ped(Calf)	1.828628	499.5096	500.5393	0.9979427	Positive
R!variance	1.000000	273.1609	410.0210	0.6662119	Positive

and the BLUP estimates by:

```

> summary(adg.asr)$coef.random

```

	solution	std error	z ratio
ped(Calf)_Sire_1	11.0252890	17.44623	0.631958069
ped(Calf)_Sire_2	-17.6385364	17.44623	-1.011022518

```

ped(Calf)_Sire_3  6.6132474  18.02693  0.366853786
ped(Calf)_Sire_4 -8.0185320  18.76570 -0.427297361
ped(Calf)_Sire_5  8.0185320  18.76570  0.427297361
ped(Calf)_Sire_6  8.4824687  17.13032  0.495172801
ped(Calf)_Sire_7  0.4445043  16.60110  0.026775595
ped(Calf)_Sire_8 -26.9640897  16.83823 -1.601361699
ped(Calf)_Sire_9  18.0371167  16.60110  1.086501425
ped(Calf)_101    -7.3121005  14.75468 -0.495578393
ped(Calf)_102    16.3994122  14.75468  1.111471922
...
ped(Calf)_164    13.7740505  14.28636  0.964140026
ped(Calf)_165     7.9907547  14.28636  0.559327589

```

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  $0.5\mathbf{I}_9$ .

A simple analysis fitting Sire and ignoring the pedigree:

```
> adg1.asr <- asreml(y1 ~ Line, random = ~ Sire, data=harvey)
```

gives

```
> summary(adg1.asr)$varcomp
              gamma component std.error  z.ratio constraint
Sire          0.2055558  27.20922  26.59081  1.023256  Positive
R!variance  1.0000000  132.36900  25.00140  5.294463  Positive
```

while including the general inverse  $0.5\mathbf{I}_9$ :

```
> sire.giv <- data.frame(row=seq(1,9),column=seq(1,9),value=rep(0.5,9))
```

```
> attr(sire.giv,"rowNames") |> paste("Sire_",seq(1,9),sep="")
```

```
> adg2.asr <- asreml(y1 ~ Line, random = ~ giv(Sire, var=T), ginverse=list(Sire=sire.giv), data=harvey)
```

gives

```
> summary(adg2.asr)$varcomp
              gamma component std.error  z.ratio constraint
giv(Sire)    0.1027814  13.61135  13.20766  1.030565  Positive
R!variance  1.0000000  132.43012  25.01868  5.293250  Positive
```

# Prediction from the linear model

## 6

### 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 generalisation 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 prediction.
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 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 Section 7.4.12. 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:

```
> rcb.asr <- asreml(yield ~ Variety, random = ~ Rep, na.method.X = "include", data = nin89)
```

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

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

A component named `predictions` is included in the `asreml` object.

```
> rcb.pv$predictions
$pvals

Notes:
- Rep terms are ignored unless specifically included
- mv is averaged over fixed levels
- Variety is included in the prediction
- (Intercept) is included in the prediction
- mv is ignored in this prediction

      Variety predicted.value standard.error est.status
1  ARAPAHOE          29.4375         3.855687 Estimable
2    BRULE           26.0750         3.855687 Estimable
3  BUCKSKIN          25.5625         3.855687 Estimable
...
54   TAM107          28.4000         3.855687 Estimable
55   TAM200          21.2375         3.855687 Estimable
56    VONA           23.6000         3.855687 Estimable

$avsed
[1] 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 arguments to `asreml()` may be included in the call to `predict.asreml()`, 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.

**mv, units** 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()` 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 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  $X$  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()` 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()` 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()` 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; 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`.
 

```

> 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.

# The asreml class

7

## 7.1 asreml

---

asreml	<i>Fit the linear mixed model</i>
--------	-----------------------------------

---

### Description

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

### Usage

```
asreml(fixed = y ~ 1, random, sparse, rcov, G.param, R.param,  
       predict = predict.asreml(), constraints = asreml.constraints(),  
       data = sys.parent(), subset, family = asreml.gaussian(),  
       weights = NULL, offset = NULL, na.method.Y = "include",  
       na.method.X = "fail", keep.order = FALSE, ran.order = "user",  
       fixgammas = FALSE, as.multivariate = NULL, model.frame = FALSE,  
       start.values = FALSE, dump.model = FALSE, model = FALSE,  
       control = asreml.control(...), ...)
```

### Arguments

<b>fixed</b>	formula object specifying the fixed effects part of the model, with the response on the left of a $\sim$ operator, and the terms, separated by + operators, on the right. If <b>data</b> is given, all names used in the formula should be defined as variables or columns in the data frame. A model with the intercept as the only fixed effect can be specified as $\sim 1$ . There must be at least one fixed effect specified. If the response evaluates to a matrix ( $y$ ) then a factor <b>trait</b> with levels <code>dimnames(y)[[2]]</code> is added to the model frame.
<b>random</b>	a formula object, specifying the random effects part of the model, with the terms, separated by + operators, on the right of a $\sim$ operator. This argument has the same general characteristics as <b>fixed</b> , but there will be no left side to the $\sim$ operator. Variance structures imposed on random terms are specified using <i>special functions</i> .
<b>sparse</b>	a formula object, specifying the fixed effects to be absorbed, with the terms, separated by + operators, on the right of a $\sim$ operator. This argument has the same general characteristics as <b>fixed</b> , but there will be no left side to the $\sim$ expression.

<code>rcov</code>	a formula object, specifying the <b>R</b> model, with the terms, separated by + operators, on the right of a ~ operator. This argument has the same general characteristics as <code>fixed</code> , but there will be no left side to the ~ expression. The default is the keyword <code>units</code> which is defined as <code>seq(1,nrow(data))</code> and automatically included in the model frame. Variance models for the residual component of the model can be specified using <i>special functions</i> .
<code>G.param</code>	a list object, generated by a call to <code>asreml.gdflt</code> using the <code>random</code> formula, holding initial parameter estimates and constraints relating to <b>G</b> . Can also be a character string, in which case it is treated as the name of a comma delimited file with a header line and a column for each variance component's name, initial value and constraint code. The internal list object is generated from the contents of this file. See <code>start.values</code>
<code>R.param</code>	a list object, generated by a call to <code>asreml.rdflt</code> using the <code>rcov</code> formula, holding initial parameter estimates and constraints, including $\sigma^2$ , relating to <b>R</b> . Can also be a character string, in which case it is treated as the name of a comma delimited file with a header line and a column for each of the variance component name, its initial value and constraint code. The required list object is generated from the contents of this file. see <code>start.values</code>
<code>predict</code>	a list object specifying the classifying factors and related options to obtain tables of predicted values from the model. This list would normally be the value returned by a call to <code>predict.asreml</code> .
<code>constraints</code>	A matrix used to constraints among the variance components with as many rows as there are (original) variance parameters and as many columns as there are constrained parameters. See <code>asreml.constraints</code> for an example.
<code>data</code>	a data frame in which to interpret the variables named in <code>fixed</code> , <code>random</code> , <code>sparse</code> , and <code>rcov</code> . If the <code>data</code> argument to <code>asreml</code> is missing, the function sets <code>data</code> to <code>sys.parent()</code> and the context for interpreting names will be the next function up the calling stack.
<code>subset</code>	a logical vector identifying which subset of the rows of the data should be used in the fit. All observations are included by default.
<code>family</code>	family object - a list of functions and expressions for defining the link and variance functions. The currently supported families are gaussian, binomial, negative binomial, poisson and Gamma. Family objects are supported via the <code>asreml</code> family functions <code>asreml.gaussian()</code> , <code>asreml.binomial()</code> etc. In addition to the link argument, these functions take an additional dispersion argument as in <code>asreml.gaussian(link="identity",dispersion=NA)</code> . The default for <code>asreml.gaussian()</code> is NA which implies that <code>asreml</code> will estimate the scale parameter, otherwise the parameter is fixed at the nominated value.
<code>weights</code>	character string or name identifying the column of <code>data</code> to use as weights in the fit.
<code>offset</code>	character string or name identifying the column of <code>data</code> to include as an offset in the model. This is ignored if <code>family=gaussian(link="identity")</code> .
<code>na.method.Y</code>	a character string (" <code>include</code> ", " <code>omit</code> " or " <code>fail</code> ") specifying how missing data in the response is to be handled. This is applied to the model frame after any <code>subset</code> argument has been used. The default (" <code>include</code> ") is to estimate missing values; this is necessary in spatial models to preserve the spatial structure. " <code>omit</code> " deletes observations that contain missing values in the response.

<code>na.method.X</code>	a character string ("include", "omit" or "fail") specifying how missing data in covariates is to be handled. This is applied to the model frame after any <code>subset</code> argument has been used but before any <code>at()</code> functions are invoked. The default "fail" causes an error if there are missing values in any covariate. "omit" deletes observations that contain missing values in any covariate.
<code>keep.order</code>	should the terms in the fixed formula be kept in the order they are specified. By default (FALSE), terms are re-ordered so that main effects appear before interactions.
<code>ran.order</code>	a character string specifying the order of terms in the <code>random</code> formula, may be one of: "user": terms are kept in the order in which they appear, "R": the order determined by <code>terms(random,keep.order=FALSE)</code> , "noef": terms ordered by increasing numbers of effects. Default is "user".
<code>fixgammas</code>	if TRUE, overrides the settings in <code>R.param</code> and <code>G.param</code> and constrains all variance parameters to be fixed.
<code>as.multivariate</code>	A character string or name specifying the column in the data that identifies the traits in a multivariate analysis. If not NULL, implies that the data for a multivariate analysis is set up as though it were for a univariate analysis.
<code>model.frame</code>	if TRUE, the model frame used in the fit is returned in the <code>asreml</code> object.
<code>start.values</code>	if TRUE, <code>asreml</code> exits prior to the fitting process and returns a list of length 3 containing the <code>G.param</code> and <code>R.param</code> lists and a data frame containing component names, initial values and boundary constraints. Initial values or constraints in the list or data frame objects can then be set.  If a character string, then a file of that name is created and the data frame object written out in comma separated form. This file can be edited externally by a text editor or spreadsheet application and subsequently used with the <code>G.param</code> or <code>R.param</code> arguments.
<code>dump.model</code>	if TRUE, <code>asreml</code> exits prior to the fitting process and returns a list with all components necessary for the fit. This argument would be used in conjunction with <code>model</code> in a simulation setting, for example, to avoid the overheads of (repeatedly) interpreting the formulae objects.
<code>model</code>	if this argument is not of mode logical, the object is assumed to have been created by the <code>dump.model</code> argument and <code>asreml</code> will extract the necessary components and perform the fit. The default is FALSE which implies normal execution; TRUE generates an error.
<code>control</code>	a list of iteration, algorithmic and parameter settings, including those related to spline knot points and known variance structures. See <code>asreml.control()</code> for their names and default values. These can also be set as argument/value pairs in the <code>asreml</code> call.

## Details

Models for `asreml` are specified symbolically in formula objects `fixed`, `random`, `sparse` and `rcov`. These objects are parsed in the context of the data frame, all internal objects required by the REML routines are constructed and a call to the underlying C and Fortran routines is generated. Variance models for terms in `random` are specified using "special" formula functions, which if not specified, defaults to (scaled) identity. See the reference guide for details on special model functions and their use. Some of these model functions require the formula arguments be partially evaluated before the final model frame is computed; for this reason, it is recommended that all names mentioned in the formulae be defined as variables in a data frame named by the `data` argument.

If the response is a matrix, a multivariate linear model is fitted to the columns of the matrix. The terms in the `fixed` formula will be re-ordered so that main effects come first, followed by the interactions, all second-order, all third-order and so on; `keep.order` can be used to modify this behaviour.

A formula has an implied intercept term. To remove this use `'y ~ -1 + ...'`. This is only effective in the `fixed` formula; in all other formula arguments any reference to the intercept is ignored.

The `subset` argument, like the terms in formula, is evaluated in the context of the data frame. The specific action of the `subset` argument is as follows: the model frame, including weights, is computed on the rows of `data` after the appropriate subset is extracted.

## Value

an object of class `asreml` representing the fit. Generic functions such as `plot()`, `predict()` and `summary()` have methods to return various results of the fit. The functions `resid()`, `coef()` and `fitted()` can be used to extract some of its components. See `asreml.object` for the components of the fit.

## Author(s)

David Butler

## References

- 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.
- Smith, A. B., Cullis, B. R., Gilmour, A.R. and Thompson, R. (1998). Multiplicative models for interaction in spatial mixed model analyses of Multi-Environment trial data, *Proceedings of the International Biometrics Conference*, Capetown.
- 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*, **48**, 269-311.

## See Also

`asreml.object` `summary.asreml` `wald.asreml` `plot.asreml` `predict.asreml` `variogram.asreml`  
`tr.asreml` `svc.asreml` `asreml.man`

## Examples

```
dat <- data.frame(y=rnorm(20),x=seq(1,20))
ex.asr <- asreml(y ~ x, data=dat)

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

## 7.2 asreml.control

asreml.control

*Set control parameters for asreml()*

## Description

Set various constants and additional arguments for `asreml`.

## Usage

```
asreml.control(knots = 50, nsppoints = 21, splinepoints = list(),
  predictpoints = list(), grid = TRUE, splinescale = -1,
  spline.step = list(dev = 10000, pol = 10000),
  pwrpoints = list(), ginverse = list(), mbf = list(),
  group = list(), Cfixed = FALSE, Csparse = formula("~NULL"),
  aom = FALSE, equate.levels = character(0), trace = TRUE,
  maxiter = 13, stepsize = 0.1, workspace = 8e+06,
  pworkspace = 8e+06, ...)
```

## Arguments

- |                            |   |
|----------------------------|---|
| <code>knots</code>         | default number of knot points for a spline. The number of knot points used is <code>min(unique( levels, knots))</code> .  |
| <code>nsppoints</code>     | influences the number of points used when predicting splines and polynomials. The design matrix generated by the <code>pol()</code> and <code>spl()</code> functions are modified to include extra rows used for prediction. The range of the data is divided by $n-1$ to give a step size $i$ . For each point $p$ in the list, a predict point is inserted at $p+i$ if there is no data value in the interval $[p, p+1.i]$ . <code>nsppoints</code> is ignored if <code>splinepoints</code> is set. This process also affects the number of levels identified by <code>dev()</code> . |
| <code>splinepoints</code>  | a list with named components where each component is a vector of user supplied knot points for a particular spline and the component name is the factor named in the <code>spl()</code> function.   |
| <code>predictpoints</code> | a list with named components where each component is a list (for two dimensions) or vector (single dimension) of user supplied predict points for <code>spl()</code> , <code>pol()</code> , <code>dev()</code> , power or Matern models. If a component of this list is in turn a list of length 2 then the first vector is taken as the $x$ coordinates and the second as the $y$ coordinates. The names of the components of the <code>predictpoints</code> list must match exactly those used in the model functions.  |
| <code>grid</code>          | a vector controlling the expansion of <code>predictpoints</code> lists for 2 dimensional kriging. For a given term, the coordinates for prediction in 2 dimensions are given as component vectors in a list argument within the <code>predictpoints</code> tree. If TRUE (the default), the $x$ and $y$ coordinates are expanded to form an $(x,y)$ grid of all possible combinations, otherwise the $x$ and $y$ vectors must be of equal length and are taken in parallel.   |
| <code>splinescale</code>   | when forming a design matrix for a <code>spl()</code> term, a standardised scale ( <code>splinescale = -1</code> ) is used. <code>splinescale = 1</code> forces ASReml to use the scale of the variable. The default is recommended in most cases.  |

<code>spline.step</code>	a list with components named <code>dev</code> and <code>pol</code> specifying the resolution for spline deviations and polynomial functions respectively. The default is <code>list(dev=10000,pol=10000)</code> . Points closer together than $1/(\text{group.step})$ of the range will be treated as a single point.
<code>pwrpoints</code>	a named list with each component containing the vector of distances to be used in a one-dimensional power model. The names of the components must match the corresponding model term named in the model formula.
<code>ginverse</code>	a named list with each component identifying a <i>G-inverse</i> matrix to include in the analysis. Each matrix can be included in the analysis as: <ol style="list-style-type: none"> <li>1. a data frame of three columns containing the non-zero elements of the lower triangle of the matrix. The first two columns of the data frame are the row and column indices and must be named <code>row</code> and <code>column</code>, respectively. This is the structure returned by <code>ASReML.Ainverse()</code>.</li> <li>2. a matrix object. Only the lower triangle is used and NAs are ignored.</li> <li>3. the complete lower triangle in vector form stored row by row; NAs are ignored.</li> </ol> in all cases every diagonal element must be present.
<code>mbf</code>	a list specifying a set of covariates to be included with one or more <code>mbf()</code> model functions. Each component of the list must in turn contain components named <code>key</code> and <code>dataFrame</code> , where <code>dataFrame</code> is a character string naming the data frame holding the covariates and <code>key</code> is a character vector of length 2 naming the columns in <code>data</code> and <code>dataFrame</code> to match the records.
<code>group</code>	a list object with named components where each component is a numeric vector specifying contiguous fields in the data frame that are to be considered as a single term. The component names can then appear in asreml model formulae.
<code>Cfixed</code>	if TRUE then the part of the C-inverse matrix that is calculated is returned in component <code>Cfixed</code> of the <code>ASReML</code> object. <code>ASReML</code> does not compute the whole C-inverse matrix but only that which is sufficient to calculate the REML solution.
<code>Csparse</code>	if not NULL then a formula object nominating the terms in the sparse component of the model for which available elements of C-inverse are required. A data frame giving the row, column and non-zero element of C-inverse for the nominated terms is returned in component <code>Csparse</code> of the <code>ASReML</code> object.
<code>aom</code>	if TRUE, invokes research results in progress on Alternate Outlier Models and returns <i>standardized conditional residuals</i> and <i>standardized conditional BLUPs</i> in the asreml object.
<code>equate.levels</code>	a character vector of factor names whose levels are to be <i>equated</i> . If factor <code>A</code> has levels <code>a,b,c,d</code> and factor <code>B</code> has levels <code>a,b,c,e</code> , the effect of <code>equate.levels</code> is that both <code>A</code> and <code>B</code> have 5 levels and <code>as.numeric(A) = 1,2,3,4</code> and <code>as.numeric(B) = 1,2,3,5</code> . See the <code>and</code> model function in the asreml reference manual.
<code>trace</code>	if TRUE then partial iteration details are displayed in the console. The component <code>monitor</code> of the <code>ASReML</code> object is a data frame containing the full convergence sequence. The <code>tr</code> method generates a graphical trace of each parameter's convergence sequence.
<code>maxiter</code>	maximum number of iterations (default is 10).
<code>stepsize</code>	update shrinkage factor. The step size actually used is <code>sqrt(stepsize)</code> .
<code>workspace</code>	size of workspace for the REML routines measured in double precision words (groups of 8 bytes). The default is <code>workspace=8e6</code> (or 64,000,000 bytes).
<code>pworkspace</code>	size of workspace for prediction from the linear model measured in double precision words (groups of 8 bytes). The default is <code>workspace=8e6</code> (or 64,000,000 bytes).

## Details

The `asreml.control()` argument/value pairs can be given directly in an `ASReML` call. These are then filtered through `asreml.control()` inside `asreml()`.

## Value

a list with components that correspond to each of the above arguments. See the `control` argument of `ASReML`.

## Author(s)

David Butler

## See Also

`asreml`

## 7.3 The asreml object

---

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

---

## Description

This class of object represents a fitted linear mixed model from the `asreml()` function. Objects of this class have methods for the generic functions `wald()`, `coef()`, `fitted()`, `plot()`, `predict()`, `residuals()`, `summary()` and `update`.

## Value

The following components must be included in a legitimate `asreml` object. The residuals, fitted values and coefficients can be extracted by the generic functions of the same name, or by the `"\$"` operator.

<b>monitor</b>	a data frame of random components (rows) by iterations (columns) tracing the convergence sequence.
<b>loglik</b>	the loglikelihood at termination.
<b>gammas</b>	a vector of <i>nk</i> variance parameter estimates from the fit.
<b>gammas.con</b>	a vector identifying the boundary constraint applied to each variance parameter ( <b>P</b> ositive, <b>F</b> ixed, <b>U</b> nconstrained).
<b>score</b>	the score vector of length number of random components.
<b>coefficients</b>	a list with three components, <b>fixed</b> , <b>random</b> and <b>sparse</b> , where the first is a vector containing the E-solutions to the mixed model equations corresponding to the fixed effects in the <i>dense</i> part of the model formula, the second is a vector containing the E-BLUPs of the random effects and the third is a vector of E-solutions corresponding to the effects in the <b>sparse</b> formula. The names of the coefficients are the same as those in the respective formulae in the call to <code>asreml()</code> with factor levels appended and separated by <code>"\_"</code> .

<code>vcoeff</code>	a list with three components, <code>fixed</code> , <code>random</code> and <code>sparse</code> , containing the unscaled variances of the coefficients. The actual variances are calculated as <code>vcoeff*object\$sigma2</code> .
<code>predictions</code>	a list object with components <code>pvals</code> , <code>sed</code> , <code>vcov</code> and <code>avsed</code> .
<code>fitted.values</code>	a vector containing the fitted values from the model, obtained by transforming the linear predictors by the inverse of the link function.
<code>linear.predictors</code>	the linear fit on link scale.
<code>residuals</code>	a vector containing the residuals from the model.
<code>hat</code>	the diagonal elements of the matrix $WC^{-1}W^T$ , the so-called <i>extended</i> hat matrix. This is the linear mixed effects model analogue of $X(X^T X)^{-1}X^T$ for ordinary linear models.
<code>sigma2</code>	the REML estimate of the scale parameter.
<code>nedf</code>	the residual degrees of freedom, <code>length(y)-rank(X)</code> .
<code>ai</code>	the inverse average information matrix of the variance parameters. A vector of length $nk(nk+1)/2$ where $nk = \text{length}(\text{gammas})$ holding the lower triangle row-wise.
<code>Cfixed</code>	reflexive generalised inverse of the coefficient matrix of the mixed model equations relating to the dense fixed effects (if <code>Cfixed=TRUE</code> , see <code>asreml.control()</code> ). The returned object is a vector containing the lower triangle row-wise of C-inverse.
<code>Csparse</code>	non-zero elements of the reflexive generalised inverse matrix of the coefficient matrix for the sparse stored model terms nominated in the <code>Csparse</code> formula (see <code>asreml.control()</code> ). The returned object is a data frame giving the row, column and non-zero element.
<code>family</code>	family object, the result of a call to <code>asreml.family</code> .
<code>call</code>	an image of the call that produced the object.
<code>license</code>	a character string containing the license information. The string has embedded new-line characters and is best formatted through <code>cat()</code> .
<code>G.param</code>	a list object containing the constraints and final estimates of the variance parameters relating to <b>G</b> , the random part of the model. This object may be used as the value of the <code>G.param</code> argument to provide initial parameter estimates to <code>asreml</code> .
<code>R.param</code>	a list object containing the constraints and final estimates of the variance parameters relating to <b>R</b> , the error structure, including $\sigma^2$ . This object may be used as the value of the <code>R.param</code> argument to provide initial parameter estimates to <code>asreml</code> for the error component of the model.

### See Also

`asreml asreml.control`

## 7.4 Methods and related functions

### 7.4.1 asreml.About

#### Description

Returns version details of the `asreml` functions and compiled code.

## Usage

```
asreml.About()
```

### 7.4.2 asreml.Ainverse

## Description

Calculates an inverse relationship matrix in sparse form from a pedigree data frame.

## Usage

```
asreml.Ainverse(pedigree, fgen = list(character(0), 0.01),
                gender = character(0), groups = 0, groupOffset = 0,
                selfing = NA, inBreed = NA, mgs = FALSE,
                mv = c("NA", "0", "*"))
```

## Arguments

<b>pedigree</b>	a data frame with (at least) three columns that correspond to the individual, male parent and female parent, respectively. The row giving the pedigree of an individual must appear before any row where that individual appears as a parent. Founders use 0 (zero) or NA in the parental columns.
<b>fgen</b>	an optional list of length 2 where <code>fgen[[1]]</code> is a character string naming the column in <code>pedigree</code> that contains the level of selfing or the level of inbreeding of an individual. In <code>pedigree[,fgen[[1]]]</code> , 0 indicates a simple cross, 1 indicates selfed once, 2 indicates selfed twice, etc. A value between 0 and 1 for a base individual is taken as its inbreeding value. If the pedigree has implicit individuals (they appear as parents but not as individuals), they will be assumed base non-inbred individuals unless their inbreeding level is set with <code>fgen[[2]]</code> where $0 < \text{fgen}[[2]] < 1$ is the inbreeding level of such individuals.
<b>gender</b>	an optional character string naming the column of <code>pedigree</code> that codes for the gender of an individual. <code>pedigree[, gender]</code> is coerced to a factor and must only have two (arbitrary) levels, the first of which is taken to mean "male". An inverse relationship matrix is formed for the X chromosome as described by Fernando and Grossman (1990) for species where the male is XY and the female is XX.
<b>groups</b>	includes genetic groups in the pedigree. The first <code>g</code> lines of the pedigree identify genetic groups (with zero in both the male and female parent columns). All other rows must specify one of the genetic groups as sire or dam if the actual parent is unknown.
<b>groupOffset</b>	Describe <code>groupOffset</code> here
<b>selfing</b>	allows for partial selfing when the third field of <code>pedigree</code> is unknown. It indicates that progeny from a cross where the male parent is unknown is assumed to be from selfing with probability $s$ and from outcrossing with probability $(1-s)$ . This is appropriate in some forestry tree breeding studies where seed collected from a tree may have been pollinated by the mother tree or pollinated by some other tree (Dutkowski and Gilmour, 2001). Do not use the <code>selfing</code> argument in conjunction with <code>inBreed</code> or <code>mgs</code> .
<b>inBreed</b>	the inbreeding coefficient for <i>base</i> individuals. This argument generates the numerator relationship matrix for inbred lines. Each cross is assumed to be selfed several times to stabilize as an inbred line as is usual for cereal crops,

for example, before being evaluated or crossed with another line. Since inbreeding is usually associated with strong selection, it is not obvious that a pedigree assumption of covariance of 0.5 between parent and offspring actually holds. The `inBreed` argument cannot be used in conjunction with `selfing` or `mgs`.

<code>mgs</code>	if TRUE, the third identity in the pedigree is the male parent of the female parent (maternal grand-sire) rather than the female parent.
<code>mv</code>	missing value indicator; elements of <code>prdigree</code> that exactly match <code>mv</code> are treated as missing.

### Details

`asreml.Ainverse` uses the method of Meuwissen and Luo (1992) to compute the inverse relationship matrix directly from the pedigree.

### Value

a list with the following components:

<code>ginv</code>	a data frame with 3 columns holding the lower triangle of the inverse relationship matrix in sparse form. The first 2 columns are the <i>row</i> and <i>column</i> indices of the matrix element, respectively, and the third column holds the (inverse) matrix element itself. Sort order is columns within rows, that is, the lower triangle row-wise. This data frame has an attribute <code>rowNames</code> containing the vector of identifiers for the rows matrix.
<code>inbreeding</code>	the inbreeding coefficient for each individual, calculated as <code>diag(A-I)</code> .
<code>det</code>	the log determinant.

### References

- Fernando, R. and Grossman, M. (1990). Genetic evaluation with autosomal and x-chromosomal inheritance, *Theoretical and Applied Genetics*, **80**, 75-80.
- Meuwissen, T. H. E., and Luo, Z. (1992) Computing inbreeding coefficients in large populations. *Genetics Selection Evolution*, **24**, 305-313.

#### 7.4.3 asreml.asUnivariate

### Description

creates a univariate data frame from a multivariate style data frame.

### Usage

```
asreml.asUnivariate(stack, data, response = "y",
                    traitName = "Trait", traitsWithinUnits = TRUE)
```

### Arguments

<code>stack</code>	a character or numeric vector specifying the columns in the multivariate data frame to be stacked, that is, the equivalent to an <code>rbind()</code> ; the remaining columns are replicated. Typically, the columns to be stacked would be a series of repeated response variates from a set of subjects.
<code>data</code>	the multivariate data frame.
<code>response</code>	a character string naming the column resulting from the stack operation; default is "y".
<code>traitName</code>	a character string naming the column in the output data frame that contains the (replicated) names of the variates in the <code>stack</code> argument; default is "Trait".
<code>traitsWithinUnits</code>	a logical response that determines the sort order of the returned data frame. If <code>TRUE</code> (the default), the returned object is sorted as traits, that is, the column identified by <code>traitName</code> , within units, where <code>units</code> is defined as <code>seq(1,nrow(data))</code> . If <code>FALSE</code> , the order is units within <code>traitName</code> .

### Details

Data frames with multivariate responses are typically arranged with a separate column for each multivariate response. In some circumstances it may be necessary to restructure the data into a univariate style, where the individual responses are arranged in a single column and an appropriate identifying column added.

### Value

a data frame with the columns identified in `stack` concatenated into a single variate named by `response`, a column `traitName` identifying the (original) response of each observation added and the remaining columns in the input data frame replicated to length. The data frame is sorted according to `traitsWithinUnits`.

#### 7.4.4 `asreml.constraints`

### Description

Return a constraints matrix that sets variance parameter constraints for `asreml`.

### Usage

```
asreml.constraints(form, gammas, drop.unused.levels = TRUE,
                  intercept = FALSE, na.action = na.include)
```

### Arguments

<code>formula</code>	a model formula including at least one factor of length $n_c$ , where $n_c$ is the number of variance parameters to be considered in the constrained set, the levels of which are taken from $\kappa$ , the full vector of variance parameters, and the number of distinct levels is less than $n_c$ .
<code>data</code>	a data frame in which to resolve the names in <code>formula</code> .

## Details

Variance parameter constraints are specified through a design matrix  $M$  from a simple linear model. Let  $\kappa$  be the  $n_k$  vector of unconstrained variance parameters and  $T$  be a  $n_k \times n_c$  imposing the linear constraints  $T^T \kappa = s$ . This is equivalent to  $\kappa = M\theta + Es$  where  $\theta$  is the  $n_c$  vector of constrained parameters.

The matrix  $M$  is given as the value to the `constraints` argument of `asreml`.  $M$  must have a `dimnames` attribute with the names of  $\kappa$  as its row names.

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

## Value

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

## References

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

### 7.4.5 asreml.man

#### Description

display the asreml reference guide in PDF form.

#### Usage

```
asreml.man(browser = "acroread")
```

#### Arguments

<code>browser</code>	not used on Windows systems. On UNIX systems, sets the appropriate PDF file viewer.
----------------------	---

### 7.4.6 asreml.read.table

#### Description

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

#### Usage

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

### Arguments

... arguments are as for `read.table()`. One of `header` or `col.names` must be specified to name the variables.

### Details

`asreml.read.table` checks if `header` or `col.names` is set and calls `read.table`. Any variable in the resulting dataframe from `read.table` that is not a factor and begins with a capital letter is converted to a factor.

### Value

a data frame with as many rows as the file has lines (or one less if `header=T`) and as many variables as the file has fields (or one less if one variable was used for row names). Fields are initially read in as character data. If all the items in a field are numeric, the corresponding variable is numeric, subject to the field naming convention below. Otherwise, it is a factor (unordered), except as controlled by the `as.is` argument to `read.table()`. All lines must have the same number of fields (except the header, which can have one less if the first field is to be used for row names). **Fields whose name begins with a capital letter are converted to factors.**

#### 7.4.7 asreml.variogram

### Description

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

### Usage

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

### Arguments

<code>x</code>	numeric vector of $x$ coordinates, may also be a matrix or data frame with 2 or 3 columns. If <code>ncol(x)</code> is 3, the columns are taken to be the $x$ and $y$ coordinates and the response ( $z$ ), respectively. If <code>ncol(x)</code> is 2, the columns are taken to be the $x$ coordinates and the response, respectively. In this case the $y$ coordinates are generated as <code>rep(1,nrow(x))</code> .
<code>y</code>	numeric vector of $y$ coordinates.
<code>z</code>	response vector.
<code>composite</code>	for data on a regular grid. If TRUE, the average of the variograms in quadrants $(x,y)$ and $(x,-y)$ is returned. Otherwise, both variograms are returned and identified as quadrants 1 and 4.
<code>model</code>	can only be "empirical" at present
<code>metric</code>	distance between $(x,y)$ points. Valid measures are "euclidean" or "manhattan".
<code>angle</code>	a vector of directions. Angles are measured in degrees anticlockwise from the $x$ axis. Default is 0.

<code>angle.tol</code>	the angle subtended by each direction. That is, an arc <code>angle +/- angle.tol/2</code> . Default is 180 which gives an omnidirectional variogram.
<code>nlag</code>	the maximum number of lags. Default is 20.
<code>maxdist</code>	the fraction of the maximum distance to include in the calculation. The default is half the maximum distance in the data.
<code>xlag</code>	the width of the lags. If missing, <code>xlag</code> is set to <code>maxdist/nlag</code> .
<code>lag.tol</code>	the distance tolerance. If missing, <code>lag.tol</code> is set to <code>xlag/2</code> .
<code>grid</code>	if <code>FALSE</code> , forces polar variograms if $(x,y)$ specifies a regular grid.

### Details

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

### Value

a data frame including the following components:

<code>x</code>	the original $x$ coordinates.
<code>y</code>	the original $y$ coordinates.
<code>gamma</code>	the variogram estimate.
<code>distance</code>	the average distance for pairs in the lag.
<code>np</code>	the number of pairs in the lag.
<code>angle</code>	direction if not a regular grid.

### References

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

#### 7.4.8 `coef.asreml`

### Description

This is a method for the function `coef()` for objects inheriting from class `asreml`. See `coef()` for the general behavior of this function.

### Usage

```
coef.asreml(object, list = FALSE, pattern = character(0))
```

### Arguments

<code>object</code>	an <code>asreml</code> object
<code>list</code>	if <code>TRUE</code> , the coefficients are returned in a named list of length the number of terms in the model.
<code>pattern</code>	an interaction term in the model, that is a character string of variable names separated by “.”, designed to extract a subset of coefficients for that term

## Value

If neither `pattern` or `list` is set, then a list of length 3 with the following components:

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

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

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

If `list=TRUE`, a list object where each component corresponds to a term in the model and is a single column matrix with a `dimnames` attribute; otherwise a single column matrix of effects as specified by `pattern`.

### 7.4.9 fitted.asreml

#### Description

Extracts fitted values from `asreml` objects.

#### Usage

```
fitted.asreml(object, type = c("response", "link"))
```

#### Arguments

`object`                an `asreml` object.

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

#### Value

The vector of fitted values.

### 7.4.10 plot.asreml

#### Description

A method for the generic function `plot()` for objects inheriting from class `asreml`.

#### Usage

```
plot.asreml(object, formula = ~NULL, fun = NULL, spatial = "plot",  
              npanels = NA, ...)
```

### Arguments

<code>object</code>	An object inheriting from class <code>asreml</code> .
<code>formula</code>	an optional formula specifying the form of a Trellis plot, where the trellis function to be used is given by the <code>fun</code> argument. This overrides the default plot behaviour. Variables in the original data frame can be referenced in the formula and the object itself can be referenced by <code>."</code> , allowing extractor functions like <code>resid</code> and <code>fitted</code> to be used.
<code>fun</code>	a character string naming a Trellis or user defined function to be used with the <code>formula</code> argument. Required if <code>formula</code> is specified.
<code>spatial</code>	If <code>"plot"</code> and an independent error has been fitted with <code>units</code> in the random formula, these are added to the residuals, otherwise if <code>"trend"</code> then <code>units</code> are not added even if present in the model.
<code>npanels</code>	an integer specifying the maximum number of panels per page. If NA then a suitable default is determined.
<code>...</code>	graphical parameters, typically title information, can be supplied to <code>plot.asreml()</code> .

### Details

By default, four plots are currently generated: a histogram of the residuals, a Normal Q-Q plot, a plot of residuals against fitted values and a plot of residuals against unit number. If the residual structure of the model contains multiple sections, the default plots are conditioned on the factor whose levels define the sections. For multivariate analyses, the plots are conditioned on `trait`.

### Value

An invisible list of trellis graph objects. The default behaviour uses `print.trellis` to position the four plots on the screen

#### 7.4.11 `plot.asrVariogram`

### Description

A Trellis plot of an `asrVariogram` object using either `xyplot` or `wireframe`.

### Usage

```
plot.asrVariogram(obj, npanels = NA, scale = TRUE, ...)
```

### Arguments

<code>obj</code>	an <code>asrVariogram</code> object from <code>variogram.asreml</code> .
<code>npanels</code>	an optional number of panels per page in multi-panel plots; if NA then a suitable default is determined.
<code>scale</code>	if TRUE and a multi-panel plot, the ordinate for each group is scaled relative to its maximum.
<code>...</code>	graphical parameters can be passed to the Trellis functions.

**Value**

An invisible Trellis object.

**7.4.12 predict.asreml****Description**

A method for the generic function `predict` for objects of class `asreml`. Forms a linear function of the vector of fixed and random effects in the linear model to obtain an estimated or predicted value.

**Usage**

```
predict.asreml(object = NULL, classify = character(0), levels = list(),
               present = list(), ignore = character(0),
               use = character(0), except = character(0),
               only = character(0), associate = formula("~NULL"),
               average = list(), vcov = logical(0), sed = logical(0),
               parallel = logical(0), aliased = logical(0), ...)
```

**Arguments**

<code>object</code>	an <code>asreml</code> object.
<code>classify</code>	a character string giving the variables that define the margins of the multiway table to be predicted. Multiway tables are specified by forming an interaction type term from the classifying variables, that is, separating the variable names with the ":" operator.
<code>levels</code>	a list, named by the margins of the classifying table, of vectors specifying the levels at which predictions are required. If omitted, factors are predicted at each level, simple covariates are predicted at their overall mean and covariates used as a basis for splines or orthogonal polynomials are predicted at their design points. Additional prediction points for spline terms should be included in the design matrix with the <code>splinepoints</code> argument (see <code>asreml.control</code> ) and included in the predict set with the <code>predictpoints</code> argument. The factors <code>mv</code> and <code>units</code> are always ignored.
<code>present</code>	a character vector specifying which variables to include in the <code>present</code> averaging set. The <code>present</code> set is used when averaging is to be based only on cells with data. The <code>present</code> set may include variables in the <code>classify</code> set but not those in the <code>average set</code> .  If a list, there can be a maximum of two components, each a character vector of variable names, representing non-overlapping <code>present</code> categorisations and one optional component named <code>prwts</code> containing a vector of weights to be used for averaging the <code>first</code> present table only. The vector(s) of names may include variables in the <code>classify</code> set but not those in the <code>average set</code> .
<code>ignore</code>	a character vector specifying which variables to ignore in the prediction process.
<code>use</code>	a character vector specifying which variables to add to the prediction model after the default rules have been invoked.
<code>except</code>	a character vector specifying which variables to exclude in the prediction process. That is, the prediction model includes all fitted model terms not in the <code>except</code> list.

<code>only</code>	a character vector specifying which variables form the prediction model, that is, the default rules are not invoked.
<code>associate</code>	a formula object specifying terms in up to two independent nested hierarchies. The factors in each hierarchy are written as a compound term separated by the ":" operator and in <i>left-to-right</i> outer to inner nesting order. Nested hierarchies are separated by the "+" operator; only one "+" operator is currently permitted, giving a maximum of two <code>associate</code> lists.
<code>average</code>	a list, named by the margins of the classifying table, specifying which variables to include in the averaging set. Optionally, each component of the list is a vector specifying the weights to use in the averaging process. If omitted, equal weights are used.
<code>vcov</code>	if TRUE, the full variance matrix of the predicted values is returned in a component <code>vcov</code> . Default is FALSE.
<code>sed</code>	if TRUE, the full standard error of difference matrix of the predicted values is returned in a component <code>sed</code> . Default is FALSE.
<code>parallel</code>	a list specifying those variables whose levels are to be expanded in parallel. Each component of the list is a vector specifying the levels in full, and all vectors must be of equal length.
<code>aliased</code>	if TRUE, the predicted values are returned for non-estimable functions. Default is FALSE.
<code>...</code>	other arguments to <code>asreml</code> can be given.

## Details

The prediction process forms 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. Prediction equations are formed just prior to the final iteration in `asreml`. The process is basically: `predict.asreml` passes a list of the user specifications to the `predict` argument of `asreml`; structures required to construct the prediction design matrix are passed to the REML routines; predicted values and standard errors are returned in a component `predictions` of the `asreml` object. `predict.asreml` calls `update.asreml` to run the model from its final solution and set the `predict` argument of `asreml`.

## Value

A list named `predictions` with the following components is included in the `asreml` object:

<code>pvals</code>	a dataframe of predicted values.
<code>sed</code>	optional matrix of standard errors of difference.
<code>vcov</code>	optional matrix of variances of predicted values.
<code>avsed</code>	summary standard error of difference.

## References

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.

### 7.4.13 residuals.asreml

#### Description

This is a method for the function `residuals` for objects inheriting from class `asreml`. See `resid()` for the general behavior of this function.

#### Usage

```
residuals.asreml(object, type = c("stdCond", "working", "deviance",  
                                "pearson", "response"),  
                 spatial = c("trend", "plot"))
```

#### Arguments

<code>object</code>	an <code>asreml</code> object
<code>type</code>	type of residuals, with choices <code>"stdCond"</code> , <code>"working"</code> , <code>"deviance"</code> , <code>"pearson"</code> , <code>"working"</code> or <code>"response"</code> ; <code>"stdCond"</code> is the default if <code>aom = TRUE</code> is given in the <code>asreml</code> call, else <code>"deviance"</code> residuals are returned unless specified otherwise.
<code>spatial</code>	if a second independent error term has been fitted by including <code>units</code> in the random formula, the residuals will have the <code>units</code> BLUPs added if <code>spatial = "plot"</code> .

#### Value

the vector of residuals.

### 7.4.14 summary.asreml

#### Description

A method for the generic function `summary` for objects inheriting from class `asreml`.

#### Usage

```
summary.asreml(object, nice = FALSE, all = FALSE)
```

#### Arguments

<code>object</code>	an <code>asreml</code> object
<code>nice</code>	if TRUE, vectors of variance parameters in lower triangular order for heterogeneous variance models are converted to full matrix form.
<code>all</code>	if TRUE, coefficients from the fixed, random and absorbed factors (sparse), and information on the error distribution are included in the returned object.

**Value**

an object of class `summary.asreml` with the following components:

<code>call</code>	the component from <code>object</code> .
<code>loglik</code>	the component from <code>object</code> .
<code>nedf</code>	the component from <code>object</code> .
<code>sigma</code>	<code>sqrt(object\$sigma2)</code> .
<code>varcomp</code>	a dataframe summarising the variance parameter vector ( <code>object\$gammas</code> ). Variance component ratios are converted to components when appropriate and a measure of precision is presented along with constraints (either as set by the user or enforced by <code>asreml</code> ).
<code>nice</code>	a list object with a component for each random term giving a <i>nice</i> form of the fitted variance model. Vectors are returned as is and matrices are converted to full form. For <code>ante</code> and <code>chol</code> models, the parameters are returned in <code>varcomp</code> and <code>nice</code> converts this to variance-covariance form.
<code>Cfixed</code>	if <code>Cfixed=TRUE</code> is specified on the call to <code>asreml</code> , the matrix partition of C-inverse for the fixed (dense) component of the model.
<code>coef.fixed</code>	a numeric matrix of E-solutions to the mixed model equations for fixed (dense) effects and their standard errors.
<code>coef.random</code>	a numeric matrix of E-BLUPs and standard errors for effects in the randoml model.
<code>coef.sparse</code>	a numeric matrix of E-solutions to the mixed model equations for fixed (sparse) effects and their standard errors.
<code>distribution</code>	a character string giving the error distribution.
<code>link</code>	a character string giving the link function
<code>deviance</code>	the component from <code>object</code>
<code>heterogeneity</code>	variance heterogeneity = <code>deviance/nedf</code> .

**7.4.15 svc.asreml****Description**

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

**Usage**

```
svc.asreml(object, order = "user")
```

**Arguments**

<code>object</code>	an object of class <code>asreml</code>
<code>order</code>	the sequence in which to consider the random terms: may be one of "user", "R" or "noef" for the order as specified in the <code>random</code> model formula, the order as determined by <code>terms(random,keep.order=FALSE)</code> or ordered by increasing number of effects, respectively.

## Details

Approximate stratum variances and degrees of freedom for simple variance components models are returned. For the linear mixed-effects model, it is often possible to consider a natural ordering of the variance component parameters, including the residual. Based on an idea due to Thompson (1980), `asreml` computes approximate stratum degrees of freedom and stratum variances by a modified Cholesky diagonalisation of the average information matrix.

## Value

A matrix of approximate stratum variances, degrees of freedom and component coefficients.

## References

Thompson, R. (1980). Maximum likelihood estimation of variance components, *Math. Operationsforsch Statistics*, **11**, 545-561.

### 7.4.16 `tr.asreml`

#### Description

Plots the updated values of nominated components from an `asreml` convergence sequence.

#### Usage

```
tr.asreml(obj, gammas = seq(along = obj$gammas),
  iter = seq(1,length(obj$monitor) - 1), loglik = FALSE,
  S2 = FALSE, Rvariance = FALSE)
```

#### Arguments

<code>obj</code>	an <code>asreml</code> object.
<code>gammas</code>	An optional integer vector of which components to include in the plots.
<code>iter</code>	An optional integer vector of iteration numbers to include.
<code>loglik</code>	If TRUE, include the loglikelihood.
<code>S2</code>	If TRUE, plot the residual variance ratio.
<code>Rvariance</code>	If TRUE, plot the residual variance component.

#### Details

`tr` extracts the `monitor` component from the object and plots a trace of the nominated components for the nominated iterations. Note that for some models the residual scale parameter may be a ratio constrained to 1.0 while for others it may be on the component scale.

### 7.4.17 `update.asreml`

#### Description

`update` extracts the call from the fitted object and evaluates that call, replacing any arguments with changed values. In particular, `G.param` and `R.param` are automatically replaced with those stored in the object.

## Usage

```
update.asreml(object, fixed., random., sparse., rcov., ...,
              evaluate = TRUE)
```

## Arguments

<code>object</code>	a valid <code>asreml</code> object with a component named <code>call</code> , the expression used to create itself.
<code>fixed.</code>	changes to the fixed formula. This is a two sided formula where "." is substituted for existing components in the <code>fixed</code> component of <code>object\$call</code> .
<code>random.</code>	changes to the random formula. This is a one sided formula where "." is substituted for existing components in the right hand side of the <code>random</code> component of <code>object\$call</code> .
<code>sparse.</code>	changes to the sparse formula. This is a one sided formula where "." is substituted for existing components in the right hand side of the <code>sparse</code> component of <code>object\$call</code> .
<code>rcov.</code>	changes to the error formula. This is a one sided formula where "." is substituted for existing components in the right hand side of the <code>rcov</code> component of <code>object\$call</code> .
<code>...</code>	additional arguments to the call, or arguments with changed values.
<code>evaluate</code>	if TRUE (the default) the new call is evaluated; otherwise the call is returned as an unevaluated expression.

## Details

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

## Value

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

## Warning

If a call to `asreml.control` exists as the value of the `control` argument in `object$call`, the `control` argument (and consequently the call to `asreml.control`) must be given in full in the call to `update.asreml` if existing arguments to `asreml.control` are to be changed or new ones added.

### 7.4.18 `variogram.asreml`

#### Description

A method to calculate the empirical variogram from an `asreml` object.

## Usage

```
variogram.asreml(object, formula = ~NULL, composite = TRUE,
                 model=c("empirical"), metric = c("euclidean",
                 "manhattan"), angle = 0, angle.tol = 180, nlag = 20,
                 maxdist = 0.5, xlag = NA, lag.tol = 0.5, grid = TRUE)
```

## Arguments

<code>object</code>	an object of class <code>asreml</code>
<code>formula</code>	an optional model formula designed to extract "residuals" from the random (G) component of the model rather than the residual (R) component. This is a two sided formula where the response is a pattern in the style required by the <code>pattern</code> argument of <code>coef.asreml</code> .
<code>composite</code>	the argument to <code>asreml.variogram</code>
<code>model</code>	the argument to <code>asreml.variogram</code>
<code>metric</code>	the argument to <code>asreml.variogram</code>
<code>angle</code>	the argument to <code>asreml.variogram</code>
<code>angle.tol</code>	the argument to <code>asreml.variogram</code>
<code>nlag</code>	the argument to <code>asreml.variogram</code>
<code>maxdist</code>	the argument to <code>asreml.variogram</code>
<code>xlag</code>	the argument to <code>asreml.variogram</code>
<code>lag.tol</code>	the argument to <code>asreml.variogram</code>
<code>grid</code>	the argument to <code>asreml.variogram</code>

## Details

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

## Value

An object of class `asrVariogram` that includes all components returned from `asreml.variogram` plus the following:

<code>group</code>	A character vector identifying any grouping structure, for example, that which would arise as a result of fitting independent error sections in <code>asreml</code> .
<code>names</code>	A list with components <code>x</code> , <code>y</code> and <code>groups</code> , each of which is a character string identifying the associated component of the dataframe.

### 7.4.19 `wald.asreml`

wald.asreml

*Wald tests for Asreml Models*

## Description

Compute either an incremental pseudo analysis of variance table using Wald statistics or conditional F-tests that respect marginality.

## Usage

```
wald.asreml(object, Ftest = formula("~NULL"),
denDF = ("none", "default", "numeric", "algebraic"),
ssType = c("incremental", "conditional"), ...)
```

## Arguments

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

## Details

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

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

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

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

## Value

A list with the following components:

**wald** an anova object if `denDF = none` and `ssType = "incremental"`, or a data frame otherwise.

**stratumVariances** If `denDF` is not `"none"`, a matrix of approximate stratum variances, degrees of freedom and component coefficients is returned for simple variance component models.

## Author(s)

David Butler

## References

Kenward, M.G. and Roger, J.H. (1997). The precision of fixed effects estimates from restricted maximum likelihood, *Biometrics*, **53**, 983-997.

## See Also

`svc.asreml`

## Examples

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

# Examples

## 8.1 Introduction

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

## 8.2 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 8.1.

Table 8.1: A split-plot field trial of oat varieties and nitrogen application

Block	Variety	Nitrogen			
		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()` 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("oats.asd",header=T)
> oats.asr <- asreml(fixed = yield ~ Variety+Nitrogen+Variety:Nitrogen,
+ random = ~ Blocks + Blocks:Wplots, 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 `Blocks` and `Blocks:Wplots` as random effects. It is not necessary to specify the residual term, which is equivalent to `Blocks:Wplots:Subplots`, as the experimental units are uniquely defined by these three factors. The fixed effects include the main effects of both `Variety` and `Nitrogen` and their interaction.

The variance components are:

```
> summary(oats.asr)$varcomp
              gamma component std.error  z.ratio constraint
Blocks          1.2111646   214.4808 168.78653  1.270722   Positive
Blocks:Wplots   0.5989373   106.0637  67.87730  1.562579   Positive
R!variance     1.0000000   177.0864  37.33342  4.743375   Positive
```

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()` returns the `gamma` and `component` values for each term in the random model, which are the variance ratio and component, respectively.

The default synopsis for testing fixed effects in `asreml()` 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    43419             245    <2e-16 ***
Variety         2     526              3    0.2264
Nitrogen       3    20020             113    <2e-16 ***
Variety:Nitrogen 6     322              2    0.9357
residual (MS)          177
```

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,

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

The incremental Wald tests have an asymptotic  $\chi^2$  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:

```
> wald(oats.asr, denDF="default")
$WaldTests

              Df denDF      F inc          Pr
(Intercept)   1     5 245.1000 0.000000e+00
Variety        2    10   1.4850 2.723869e-01
Nitrogen       3    45  37.6900 4.034452e-08
Variety:Nitrogen 6    45   0.3028 9.321988e-01

$stratumVariances

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

This is a simple problem for balanced designs, such as the split plot design, but it is not straightforward to determine the relevant denominator df in unbalanced designs, such as the rat data set described in the next section.

Tables of predicted means for the **Variety**, **Nitrogen** and **Variety:Nitrogen** effects can be obtained from the **predict** method:

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

This returns the usual **asreml** object in **oats.pv** with an additional component named **predictions** that has components for the predicted means for each member of the classify list as well as the full matrix of SEDs for the **Variety:Nitrogen** table.

```
> oats.pv$predictions
$Nitrogen
$Nitrogen$pvals
```

Notes:

- Variety is averaged over fixed levels
- Blocks terms are ignored unless specifically included
- Wplots terms are ignored unless specifically included
- The cells of the hypertable are calculated from all model terms constructed solely from factors in the averaging and classify sets.

```

      Nitrogen predicted.value standard.error est.status
1  0.2_cwt      98.88889      7.17471  Estimable
2  0.4_cwt     114.22222      7.17471  Estimable
3  0.6_cwt     123.38889      7.17471  Estimable
4   0_cwt      79.38889      7.17471  Estimable

```

```

$Nitrogen$avsed
[1] 4.435755

```

```

$Variety
$Variety$pvals

```

Notes:

- Nitrogen is averaged over fixed levels
- Blocks terms are ignored unless specifically included
- Wplots terms are ignored unless specifically included
- The cells of the hypertable are calculated from all model terms constructed solely from factors in the averaging and classify sets.

```

      Variety predicted.value standard.error est.status
1 Golden_rain      104.5000      7.797539  Estimable
2 Marvellous      109.7917      7.797539  Estimable
3  Victory        97.6250      7.797539  Estimable

```

```

$Variety$avsed
[1] 7.078904

```

```

$`Variety:Nitrogen`
$`Variety:Nitrogen`$pvals

```

Notes:

```

      Variety Nitrogen predicted.value standard.error est.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

```

```

$`Variety:Nitrogen`$sed
      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8]
[1,]      NA 7.682954 7.682954 7.682954 9.715025 9.715025 9.715025 9.715025
[2,] 7.682954      NA 7.682954 7.682954 9.715025 9.715025 9.715025 9.715025
[3,] 7.682954 7.682954      NA 7.682954 9.715025 9.715025 9.715025 9.715025
[4,] 7.682954 7.682954 7.682954      NA 9.715025 9.715025 9.715025 9.715025
[5,] 9.715025 9.715025 9.715025 9.715025      NA 7.682954 7.682954 7.682954
[6,] 9.715025 9.715025 9.715025 9.715025 7.682954      NA 7.682954 7.682954
[7,] 9.715025 9.715025 9.715025 9.715025 7.682954 7.682954      NA 7.682954
[8,] 9.715025 9.715025 9.715025 9.715025 7.682954 7.682954 7.682954      NA

```

```

[9,] 9.715025 9.715025 9.715025 9.715025 9.715025 9.715025 9.715025 9.715025
[10,] 9.715025 9.715025 9.715025 9.715025 9.715025 9.715025 9.715025 9.715025
[11,] 9.715025 9.715025 9.715025 9.715025 9.715025 9.715025 9.715025 9.715025
[12,] 9.715025 9.715025 9.715025 9.715025 9.715025 9.715025 9.715025 9.715025
      [,9]    [,10]    [,11]    [,12]
[1,] 9.715025 9.715025 9.715025 9.715025
[2,] 9.715025 9.715025 9.715025 9.715025
[3,] 9.715025 9.715025 9.715025 9.715025
[4,] 9.715025 9.715025 9.715025 9.715025
[5,] 9.715025 9.715025 9.715025 9.715025
[6,] 9.715025 9.715025 9.715025 9.715025
[7,] 9.715025 9.715025 9.715025 9.715025
[8,] 9.715025 9.715025 9.715025 9.715025
[9,]          NA 7.682954 7.682954 7.682954
[10,] 7.682954          NA 7.682954 7.682954
[11,] 7.682954 7.682954          NA 7.682954
[12,] 7.682954 7.682954 7.682954          NA

$`Variety:Nitrogen`$saved
      min      mean      max
7.682954 9.160824 9.715025

```

For the first two predictions, the average SED is calculated from the average variance of differences.

### 8.3 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 8.2.

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 **asrem1()** call is:

```
> rats.asr <- asrem1(weight ~ littersize+Dose+Sex+Dose:Sex, random = ~ Dam, data = rats)
```

The abbreviated output from **asrem1()** convergence monitoring, followed by variance component (from **summary()**) and Wald tests (from **wald()**) tables are:

```

> rats.asr$monitor[,-2:-5]

      1          6          7      final constraint
loglik  74.2174175  87.2397736  87.2397915  87.2397915      <NA>
S2      0.1967003  0.1653213  0.1652993  0.1652993      <NA>
df      315.0000000 315.0000000 315.0000000 315.0000000      <NA>
Dam     0.1000000  0.5854392  0.5866881  0.5866742  Positive
R!variance 1.0000000  1.0000000  1.0000000  1.0000000  Positive
>

```

Table 8.2: 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	

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

          gamma component std.error  z.ratio constraint
Dam 0.5866742 0.09697687 0.03318527  2.922287  Positive
R!variance 1.0000000 0.16529935 0.01367005 12.092083  Positive

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

$Wald
      Df denDF    F_inc    F_con Margin      Pr
(Intercept) 1  32.0 9049.0000 1099.0000      0.000000e+00
littersize  1  31.5  27.9900  46.2500      B 1.690248e-07
Dose       2  23.9  12.1500  11.5100      A 3.132302e-04
Sex       1 299.8  57.9600  57.9600      A 0.000000e+00
Dose:Sex   2 302.1   0.3984   0.3984      B 6.733474e-01

$stratumVariances
      df Variance    Dam R!variance
Dam   22.56348 1.2776214 11.46995      1
R!variance 292.43652 0.1652996 0.00000      1
>
```

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

```
> plot(rats.asr, formula=resid(.) fitted(.), fun="xyplot")
```

is shown in Figure 8.1. 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.

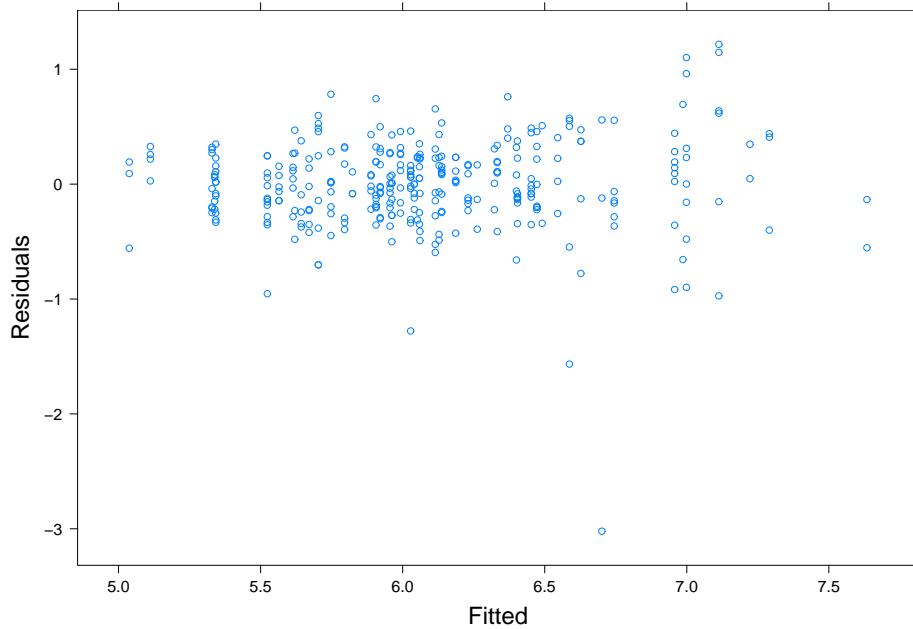


Figure 8.1: Residual plot for the rat data

We refit the model without the `Dose:Sex` term.

```
> rats2.asr <- asreml(weight ~ littersize+Sex+Dose, random = ~ Dam, data = rats)
```

```
> summary(rats2.asr)$varcomp
```

```

      gamma component std.error  z.ratio constraint
      Dam 0.595157 0.09791776 0.03341462  2.930386   Positive
R!variance 1.000000 0.16452427 0.01356057 12.132547   Positive

```

```
> wald(rats2.asr,denDF="default",ssType="conditional")
```

```

$Wald
      Df denDF  F_inc  F_con Margin      Pr
(Intercept) 1  32.0 8981.00 1093.00      0.000000e+00
littersize  1  31.4  27.85  46.43      A 1.643469e-07
Dose        2  24.0  12.05  11.42      A 3.278549e-04
Sex         1 301.7  58.27  58.27      A 0.000000e+00

```

```
$stratumVariances
```

```

      df Variance      Dam R!variance
Dam    22.60301 1.2878697 11.47231      1
R!variance 294.39699 0.1645245 0.00000      1

```

```
>
```

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:

```
> rats3.asr <- asreml(weight littersize+Dose+Sex,data = rats)
> wald(rats3.asr,denDF="default",ssType="conditional")

$Wald
      Df denDF    F_inc  F_con Margin      Pr
(Intercept)  1   317 47080.00 3309.00      0.000000e+00
littersize   1   317   68.48  146.50      A 0.000000e+00
Dose         2   317   60.99   58.43      A 0.000000e+00
Sex          1   317   24.52   24.52      A 2.328158e-06

$stratumVariances
NULL
>
```

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.

## 8.4 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.asr <- asreml(voltage ~ 1,
+ random = ~ Setstat+Setstat:Regulatr+ Teststat+Setstat:Teststat, 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

      gamma  component  std.error  z.ratio  constraint
Setstat      2.334163e-01 1.193692e-02 8.814339e-03 1.354262  Positive
Setstat:Regulatr 6.018174e-01 3.077697e-02 8.453330e-03 3.640810  Positive
Teststat      6.427520e-02 3.287037e-03 3.337300e-03 0.984939  Positive
Setstat:Teststat 1.011929e-07 5.175009e-09 5.323475e-10 9.721111  Boundary
R!variance      1.000000e+00 5.114004e-02 5.260720e-03 9.721111  Positive
```

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()` 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:

```
> temp <- asreml.gammas.ed(voltage.asr)
#
# Edit appropriate parameter code
#
> voltage.asr <- asreml(voltage ~ 1,
+ random = ~ Setstat+Setstat:Regulatr+ Teststat+Setstat:Teststat,
+ 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 8.2). 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].

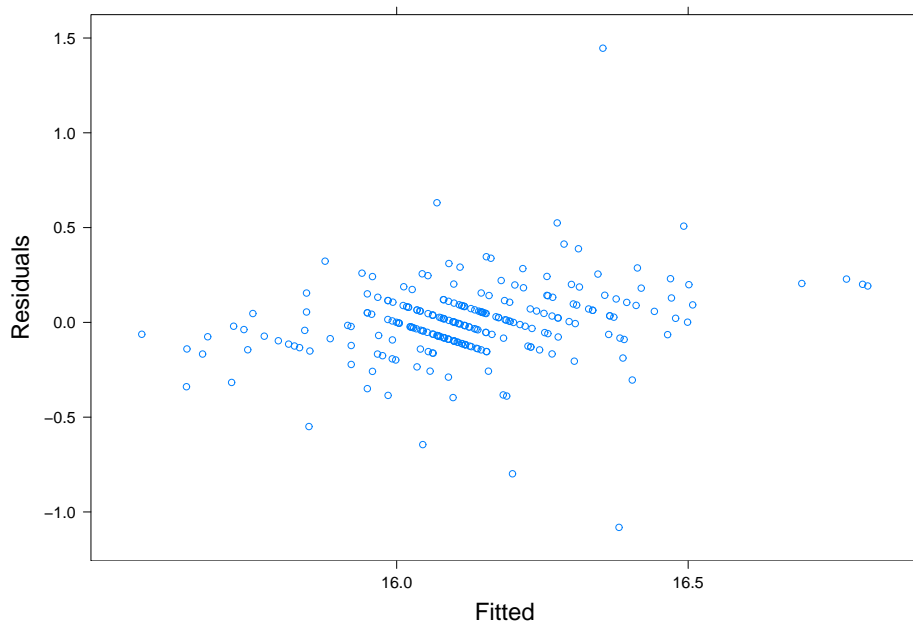


Figure 8.2: Residuals vs fitted values for the voltage data

The model omitting the `Setstat:Teststat` term:

```
> voltage2.asr <- asreml(voltage ~ 1,
+ random = ~ Setstat+Setstat:Regulatr+Teststat, data = voltage)
```

returns a REML log-likelihood of 203.242 - the same as the REML log-likelihood for the previous model. The summary of the variance components for this model are

```
> summary(voltage2.asr)$varcomp
```

	gamma	component	std.error	z.ratio	constraint
Setstat	0.23341667	0.011936975	0.008813969	1.3543246	Positive
Setstat:Regulatr	0.60181723	0.030777054	0.008453248	3.6408553	Positive
Teststat	0.06427521	0.003287047	0.003337314	0.9849378	Positive
R!variance	1.00000000	0.051140201	0.005260744	9.7210963	Positive

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 8.3).

Table 8.3: REML log-likelihood ratio test for each variance component in the voltage data

term	loglikelihood	$-2 \times$ difference	P-value
– Setstat	200.31	5.864	.0077
– Setstat:Regulatr	184.15	38.19	.0000
– Teststat	199.71	7.064	.0039

## 8.5 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 8.3.

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
MAV     1 21.0 39.7 47.0 53.0 55.0
MAV     2 32.0 59.5 63.5 65.0 67.6
MAV     3 35.5 54.6 58.0 61.5 61.5
MAV     4 33.5 41.0 48.0 57.0 58.0
MAV     5 31.5 45.3 62.0 104.0 104.0
MAV     6 27.0 43.3 56.4 74.5 62.0
MAV     7 37.0 53.0 63.0 70.3 75.9
HC      8 28.5 47.0 54.7 55.5 57.0
HC      9 48.0 62.7 106.0 125.5 123.5
```

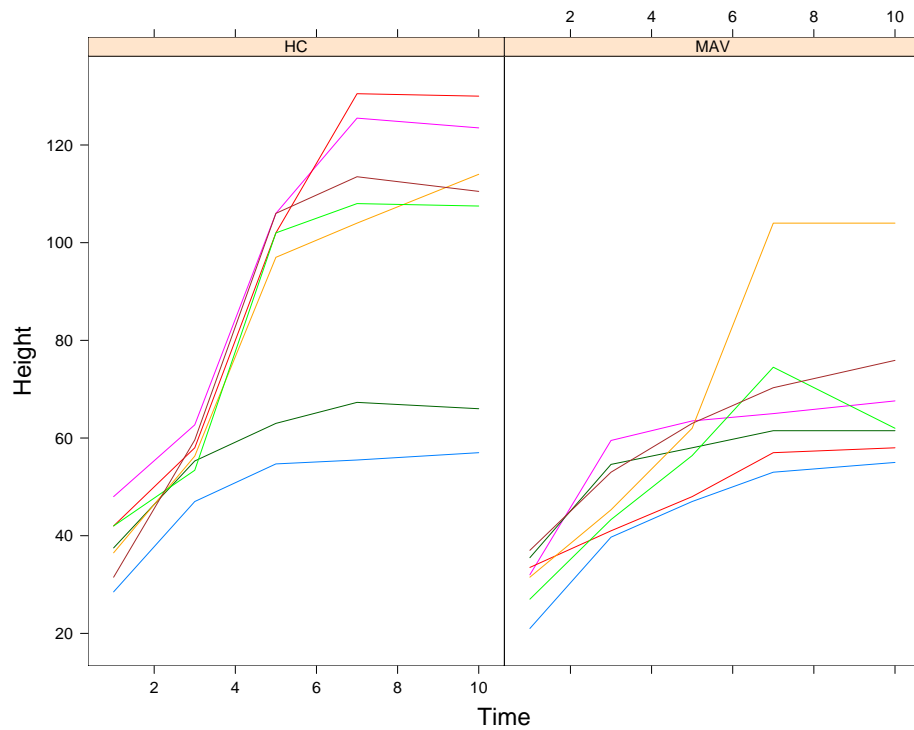


Figure 8.3: Trellis plot of plant height for each of 14 plants

```

HC  10 37.5 55.3 63.0 67.3 66.0
HC  11 42.0 58.0 102.0 130.5 130.0
HC  12 36.5 56.3 97.0 104.0 114.0
HC  13 42.0 53.4 102.0 108.0 107.5
HC  14 31.5 59.6 106.0 113.5 110.5

```

while `grassUV` is in *univariate* form:

```
> grassUV
```

```

Tmt Plant Time HeightID    y
MAV    1    1      y1  21.0
MAV    1    3      y3  39.7
MAV    1    5      y5  47.0
MAV    1    7      y7  53.0
MAV    1   10     y10 55.0
MAV    2    1      y1  32.0
MAV    2    3      y3  59.5
...
HC     13   10     y10 107.5
HC     14    1      y1  31.5
HC     14    3      y3  59.6
HC     14    5      y5 106.0
HC     14    7      y7 113.5
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

$$\mathbf{Y} = \mathbf{DT} + \mathbf{E} \quad (8.1)$$

where  $\mathbf{Y}^{14 \times 5}$  is the matrix of heights,  $\mathbf{D}^{14 \times 2}$  is the design matrix,  $\mathbf{T}^{2 \times 5}$  is the matrix of fixed effects and  $\mathbf{E}^{14 \times 5}$  is the matrix of errors. The heights taken on the same plants will be correlated and so we assume that

$$\text{var}(\text{vec}(\mathbf{E})) = \mathbf{I}_{14} \otimes \boldsymbol{\Sigma} \quad (8.2)$$

where  $\boldsymbol{\Sigma}^{5 \times 5}$  is a symmetric positive definite matrix.

The variance models used for  $\boldsymbol{\Sigma}$  are summarised in Table 8.4. These 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()` 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()` in the same order, that is **trait nested within experimental units**. In this case `asreml()` 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 `rcov` formula that define 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()` checks the sort order and reports an error if incorrect.

#### Uniform

```
> grass.asr <- asreml(cbind(y1,y3,y5,y7,y10) ~ trait+Tmt+trait:Tmt,
+ random = ~ units, rcov = ~ units:trait, data = grass)
```

#### Power

```
> grass2.asr <- asreml(y ~ Tmt+Time+Tmt:Time,
+ rcov = ~ Plant:exp(Time), data = grassUV)
```

#### Heterogeneous power

```
> grass3.asr <- asreml(y ~ Tmt+Time+Tmt:Time,
+ rcov = ~ Plant:exp(Time), data = grassUV)
```

#### Antedependence

```
> grass4.asr <- asreml(cbind(y1,y3,y5,y7,y10) ~ trait+Tmt+trait:Tmt,
+ rcov = ~ units:ante(trait), data = grass)
```

#### Unstructured

```
> grass5.asr <- asreml(cbind(y1,y3,y5,y7,y10) ~ trait+Tmt+trait:Tmt,
+ rcov = ~ units:us(trait), data = grass)
```

Table 8.4: Summary of variance models fitted to the plant data

model	number of parameters	REML loglikelihood	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,
 

```
> grass1.asr <- asreml(cbind(y1,y3,y5,y7,y10) ~ trait+Tmt+trait:Tmt,
+ rcov = ~ units:cor(trait), data = grass)
```
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.

where 1 and 3 are equivalent as are 2 and 4. The two forms for  $\Sigma$  are given by

$$\Sigma = \sigma_1^2 \mathbf{J} + \sigma_2^2 \mathbf{I}, \quad \text{units} \quad (8.3)$$

$$\Sigma = \sigma_e^2 \mathbf{I} + \sigma_e^2 \rho (\mathbf{J} - \mathbf{I}), \quad \text{cor()} \quad (8.4)$$

It follows that

$$\sigma_e^2 = \sigma_1^2 + \sigma_2^2 \quad (8.5)$$

$$\rho = \frac{\sigma_1^2}{\sigma_1^2 + \sigma_2^2} \quad (8.6)$$

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 (8.6).

```
> summary(grass.asr)$loglik
[1] -196.8768
> summary(grass.asr)$varcomp
           gamma component std.error  z.ratio constraint
units      1.263422  159.8157  75.74879  2.109812   Positive
R!variance 1.000000  126.4943  25.82054  4.898979   Positive
```

```
> summary(grass1.asr)$loglik
[1] -196.8768
> summary(grass1.asr)$varcomp
           gamma component std.error  z.ratio constraint
R!variance 1.000000  286.3098952  78.3448584  3.654482   Positive
R!trait.cor 0.5581911   0.5581911  0.1303821  4.281196 Unconstrained
```

A more plausible model for repeated measures data would allow the correlations to decrease as the lag increases. The simplest model often used which 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 in weeks. The variance parameters from this model are:

```
> summary(grass2.asr)$varcomp
           gamma component std.error  z.ratio constraint
R!variance 1.000000  301.3581311  96.67407884  3.117259   Positive
R!Time.pow 0.9190129   0.9190129  0.03117151  29.482466 Unconstrained
```

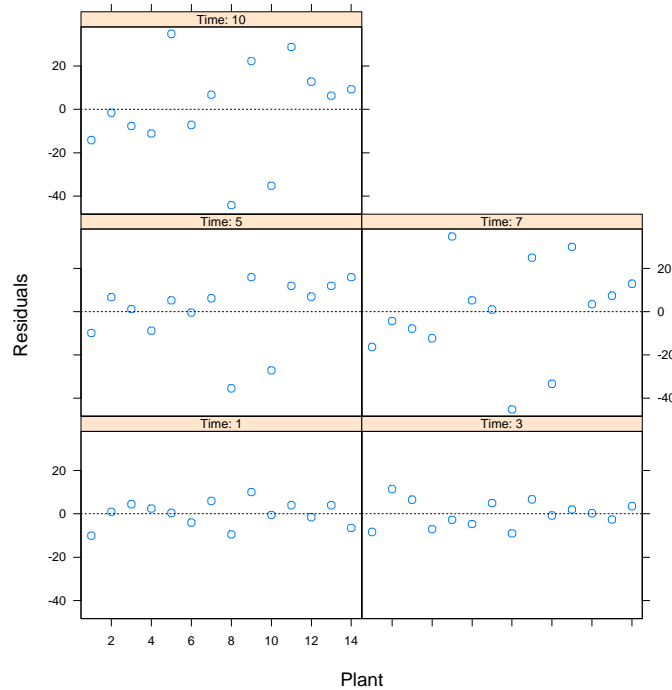


Figure 8.4:  $\text{residual} \sim \text{Plant} \mid \text{Time}$  for the  $\text{exp}()$  variance model for the plant data

When fitting such models be careful to ensure the scale of the defining variate, here `time`, does not result in an estimate of  $\phi$  too close to 1. For example, use of days in this example would result in an estimate for  $\phi$  of about 0.993.

```
> plot(grass2.asr, formula = resid(.) ~ Plant | Time, fun="xyplot")
```

creates a trend plot (Figure 8.4) of residuals against the factors that index the experimental units.

The residual plot suggests increasing variance over time. This can be modelled via the  $\text{exp}()$  variance function, which models  $\Sigma$  by

$$\Sigma = D^{0.5} C D^{0.5}$$

where  $D$  is a diagonal matrix of variances and  $C$  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
```

	gamma	component	std.error	z.ratio	constraint
R!variance	1.000000	1.000000	NA	NA	Fixed
R!Time.pow	0.906702	0.906702	0.04156582	21.813643	Unconstrained
R!Time.1	60.716256	60.716256	28.40226409	2.137726	Positive
R!Time.3	73.266300	73.266300	36.98538578	1.980953	Positive
R!Time.5	308.521040	308.521040	138.29720253	2.230855	Positive
R!Time.7	435.122455	435.122455	172.05226788	2.529013	Positive
R!Time.10	381.527027	381.527027	138.94461658	2.745893	Positive

Note that `asrem()` fixes the scale parameter to 1 to ensure that the elements of  $D$  are identifiable. The final two models considered are the antedependence model of order 1 and the unstructured

model. Both require as starting values the lower triangle of the full variance matrix. By default, `asrem1()` 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  $\Sigma$  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  $\Sigma$  by the inverse cholesky decomposition

$$\Sigma = UDU'$$

where  $D$  is a diagonal matrix and  $U$  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  $D$  and 4 in  $U$ . 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  $D$  and then the row of  $U$ :

```
> summary(grass4.asr)$varcomp
```

	gamma	component	std.error	z.ratio	constraint
R!variance	1.000000000	1.000000000	NA	NA	Fixed
R!trait.y1:y1	0.026862013	0.026862013	0.011023470	2.436802	Unconstrained
R!trait.y3:y1	-0.628357272	-0.628357272	0.246074475	-2.553525	Unconstrained
R!trait.y3:y3	0.037296080	0.037296080	0.015467150	2.411309	Unconstrained
R!trait.y5:y3	-1.490928182	-1.490928182	0.586337948	-2.542780	Unconstrained
R!trait.y5:y5	0.005994700	0.005994700	0.002468106	2.428867	Unconstrained
R!trait.y7:y5	-1.280440740	-1.280440740	0.206796644	-6.191787	Unconstrained
R!trait.y7:y7	0.007896965	0.007896965	0.003232797	2.442765	Unconstrained
R!trait.y10:y7	-0.967801877	-0.967801877	0.062806208	-15.409335	Unconstrained
R!trait.y10:y10	0.039063461	0.039063461	0.015947584	2.449491	Unconstrained

Finally, the estimated components for the unstructured model using default starting values.

```
> summary(grass5.asr)$varcomp
```

	gamma	component	std.error	z.ratio	constraint
R!variance	1.00000	1.00000	NA	NA	Fixed
R!trait.y1:y1	37.22619	37.22619	15.19745	2.449503	Unconstrained
R!trait.y3:y1	23.39345	23.39345	13.20621	1.771398	Unconstrained
R!trait.y3:y3	41.51952	41.51952	16.95023	2.449496	Unconstrained
R!trait.y5:y1	51.65238	51.65238	32.03335	1.612456	Unconstrained
R!trait.y5:y3	61.91690	61.91690	34.87103	1.775597	Unconstrained
R!trait.y5:y5	259.12143	259.12143	105.78295	2.449558	Unconstrained
R!trait.y7:y1	70.81131	70.81131	46.13709	1.534802	Unconstrained
R!trait.y7:y3	57.61452	57.61452	46.74100	1.232634	Unconstrained
R!trait.y7:y5	331.80679	331.80679	145.19689	2.285220	Unconstrained
R!trait.y7:y7	551.50690	551.50690	225.14337	2.449581	Unconstrained
R!trait.y10:y1	73.78571	73.78571	46.21175	1.596687	Unconstrained
R!trait.y10:y3	62.56905	62.56905	46.92608	1.333353	Unconstrained
R!trait.y10:y5	330.85060	330.85060	144.31864	2.292501	Unconstrained
R!trait.y10:y7	533.75583	533.75583	220.57976	2.419786	Unconstrained
R!trait.y10:y10	542.17548	542.17548	221.33410	2.449580	Unconstrained

The antedependence model of order 1 is clearly the more parsimonious model (Table 8.4). There is a surprising level of discrepancy between models for the Wald tests (Table 8.5). The main effect of treatment is significant for the uniform, power and antedependence models.

Table 8.5: 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
antependence (order 1)	4.19	15.63
unstructured	1.72	17.86

## 8.6 Spatial analysis of a field experiment

This section illustrates spatial and incomplete block analyses of a field experiment using `asreml()`. 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  $10 \times 15$  rectangular grid. Table 8.6 shows the layout of the experiment and the coding of the replicates and lattice blocks. The columns in the data frame are:

```
> shf <- asreml.read.table("barley.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.

Three models are considered: two spatial and the traditional lattice analysis 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 ( $\mathbf{R}$ -structure) is given by

$$\sigma^2 \mathbf{\Sigma} = \sigma^2 (\mathbf{\Sigma}_c \otimes \mathbf{\Sigma}_r) \quad (8.7)$$

where  $\mathbf{\Sigma}_c$  and  $\mathbf{\Sigma}_r$  are  $15 \times 15$  and  $10 \times 10$  matrix functions of the column ( $\phi_c$ ) and row ( $\phi_r$ ) autoregressive parameters respectively.

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:

```
> barley1.asr <- asreml(yield ~ Variety, rcov = ~ ar1(Row):ar1(Column), data=shf)
```

Table 8.6: 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

The REML log-likelihood, random components and Wald statistics from the fit are:

```
> summary(barley1.asr)$loglik
[1] -700.3225
> summary(barley1.asr)$varcomp

           gamma    component    std.error    z.ratio    constraint
R!variance  1.0000000 3.873880e+04 7.747479e+03 5.000181    Positive
R!Row.ar1   0.4585092 4.585092e-01 8.259184e-02 5.551507 Unconstrained
R!Column.ar1 0.6837766 6.837766e-01 6.329681e-02 10.802701 Unconstrained

> 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  33004556             852 < 2.2e-16 ***
```

```
Variety      24 12119586          313 < 2.2e-16 ***
residual (MS)      38739
```

```
> plot(variogram(barley1.asr))
```

plots the sample variogram shown in Figure 8.5.

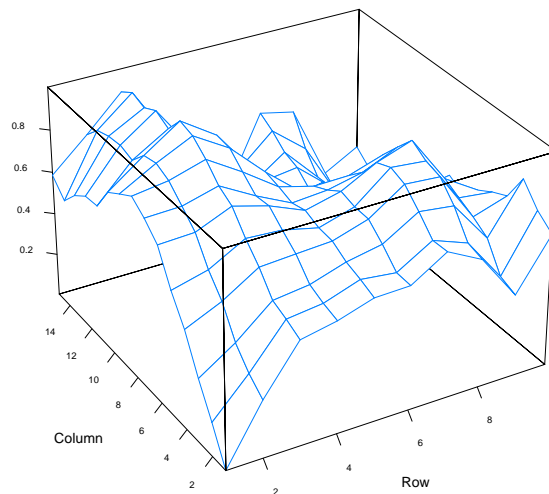


Figure 8.5: Sample variogram of the AR1×AR1 model for the Slate Hall data

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.

An extension to this model includes a measurement error or *nugget effect* term:

```
> barley2.asr <- asreml(yield ~ Variety, random = ~ units,
+ rcov = ~ ar1(Row):ar1(Column),data=shf)
```

That is, the variance model for the plot errors is now given by

$$\sigma^2 \Sigma = \sigma^2 (\Sigma_c \otimes \Sigma_r) + \psi \mathbf{I}_{150} \quad (8.8)$$

where  $\psi$  is the ratio of nugget variance to error variance ( $\sigma^2$ ). The results show a significant improvement in the REML log-likelihood with the inclusion of the nugget effect (Table 8.7).

```
> summary(barley2.asr)$loglik
[1] -696.8227
```

```
> summary(barley2.asr)$varcomp
```

```

              gamma    component    std.error    z.ratio    constraint
units        0.1061724 4.859930e+03 1.787849e+03 2.718311    Positive
R!variance   1.0000000 4.577396e+04 1.667323e+04 2.745357    Positive
R!Row.ar1    0.6826403 6.826403e-01 1.022940e-01 6.673317    Unconstrained
R!Column.ar1 0.8437888 8.437888e-01 6.848144e-02 12.321425    Unconstrained

```

```

> 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  11918530           260 < 2.2e-16 ***
Variety       24  11237959           246 < 2.2e-16 ***
residual (MS)          45774

```

The lattice analysis (with recovery of inter-block information) is:

```
> barley3.asr <- asreml(yield ~ Variety, random = ~ Rep + RowBlk + ColBlk, data=shf)
```

```

> summary(barley3.asr)$loglik
[1] -707.7857

```

```

> summary(barley3.asr)$varcomp
              gamma component std.error    z.ratio constraint
Rep          0.5287136  4262.361  6886.823 0.6189155    Positive
RowBlk       1.9344361 15594.957  5090.787 3.0633685    Positive
ColBlk       1.8372487 14811.456  4865.667 3.0440754    Positive
R!variance   1.0000000  8061.759  1340.449 6.0142228    Positive

```

```

> 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   9808702           1217 < 2.2e-16 ***
Variety       24   1711179            212 < 2.2e-16 ***
residual (MS)          8062

```

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 lattice analysis (Table 8.7). We note that the Wald statistic for the spatial model including the nugget effect is smaller than that for the AR1×AR1 model.

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$predictions$Variety$pvals
```

Table 8.7: 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

```
Variety predicted.value standard.error estStatus
1      1      1257.981      64.61878 Estimable
2      2      1501.442      64.98267 Estimable
3      3      1404.987      64.63038 Estimable
4      4      1412.569      64.90703 Estimable
5      5      1514.480      65.59318 Estimable
...
23     23      1311.490      64.07718 Estimable
24     24      1586.785      64.70481 Estimable
25     25      1592.021      63.59445 Estimable
```

```
> barley1.pv$predictions$Variety$saved
[1] 59.05192
```

```
> barley2.pv <- predict(barley2.asr, classify="Variety")
```

```
> barley2.pv$predictions$Variety$pvals
```

```
Variety predicted.value standard.error estStatus
1      1      1245.582      97.87621 Estimable
2      2      1516.234      97.86434 Estimable
3      3      1403.984      98.25699 Estimable
4      4      1404.918      98.00456 Estimable
5      5      1471.612      98.37778 Estimable
...
23     23      1316.874      98.05743 Estimable
24     24      1557.522      98.14444 Estimable
25     25      1573.888      97.99763 Estimable
```

```
> barley2.pv$predictions$Variety$saved
[1] 60.51085
```

```
> barley3.pv <- predict(barley3.asr, classify="Variety")
```

```
> barley3.pv$predictions$Variety$pvals
```

```
Variety predicted.value standard.error estStatus
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
...
23     23      1329.109      60.1994 Estimable
```

```
24      24      1546.470      60.1994 Estimable
25      25      1630.629      60.1994 Estimable
```

```
> barley3.pv$predictions$Variety$saved
[1] 62.01934
```

Notice the differences in SEs and SEDs associated with the various models. Choosing a model on the basis of smallest SE or SED is not recommended because the model is not necessarily fitting the variability present in the data.

## 8.7 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("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.

```
> wheat1.asr <- asreml(yield ~ weed, random = ~ Variety,
+ rcov = ~ ar1(Row):Column, data = wheat)
```

```
> summary(wheat1.asr)$loglik
[1] -4239.88
```

```
> summary(wheat1.asr)$varcomp
```

	gamma	component	std.error	z.ratio	constraint
Variety	0.9594791	8.279572e+04	9.217376e+03	8.982569	Positive
R!variance	1.0000000	8.629236e+04	9.462026e+03	9.119861	Positive
R!Row.ar1	0.6723405	6.723405e-01	4.184392e-02	16.067817	Unconstrained

The REML estimate of the autoregressive parameter indicates substantial within column heterogeneity.

A two dimensional spatial model is fitted with:

```
> wheat2.asr <- asreml(yield ~ weed, random = ~ Variety,
+ rcov = ~ ar1(Row):ar1(Column), data = wheat)
```

```
> summary(wheat2.asr)$loglik
[1] -4233.647
```

```
> summary(wheat2.asr)$varcomp
```

	gamma	component	std.error	z.ratio	constraint
Variety	1.0603771	8.811748e+04	8.884899e+03	9.917669	Positive
R!variance	1.0000000	8.310014e+04	9.340520e+03	8.896736	Positive
R!Row.ar1	0.6853871	6.853871e-01	4.115303e-02	16.654595	Unconstrained
R!Column.ar1	0.2859093	2.859093e-01	7.390416e-02	3.868650	Unconstrained

The change in REML log-likelihood is significant ( $\chi_1^2 = 12.46$ ,  $P < 0.001$ ) with the inclusion of the autoregressive parameter for `Column`. The sample variogram of the residuals for the  $\text{ar1} \times \text{ar1}$  model, Figure 8.6, indicates a linear drift from column 1 to column 10. We include a linear regression coefficient `pol(Column,-1)` in the model to account for this.

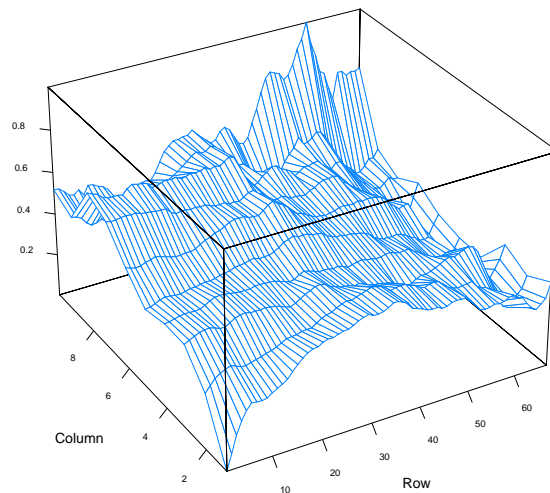


Figure 8.6: Sample variogram of the  $\text{AR1} \times \text{AR1}$  model for the Tullibigeal data

```
> wheat3.asr <- asreml(yield ~ weed + pol(Column,-1), random = ~ Variety,
+ rcov = ~ ar1(Row):ar1(Column), data = wheat)
```

Note we use the '-1' option in the `pol()` function to exclude the overall constant in the regression, as it is already fitted.

```
> summary(wheat3.asr)$loglik
[1] -4225.631
```

```
> summary(wheat3.asr)$varcomp
```

	gamma	component	std.error	z.ratio	constraint
Variety	1.1436952	8.898632e+04	8.976677e+03	9.913058	Positive
R!variance	1.0000000	7.780597e+04	8.854452e+03	8.787215	Positive
R!Row.ar1	0.6714360	6.714360e-01	4.287844e-02	15.659058	Unconstrained
R!Column.ar1	0.2660882	2.660882e-01	7.541536e-02	3.528303	Unconstrained

```
> wald(wheat3.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	551060049	7082	< 2.2e-16 ***
weed	1	7155306	92	< 2.2e-16 ***
pol(Column, -1)	1	679668	9	0.003121 **
residual (MS)		77806		

```
> summary(wheat3.asr)$coef.fixed
```

	solution	std error	z ratio
pol(Column, -1)	-139.5638	47.22053	-2.955575
weed	-182.7066	21.83804	-8.366439
(Intercept)	2872.7366	34.82716	82.485524

The linear regression of column number on yield is significant (Wald statistic = 8.74). The sample variogram (Figure 8.7) 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:

```
> wheat4.asr <- asreml(yield ~ pol(Column,-1), random = ~ Variety + units, sparse = ~ weed,
+ rcov = ~ ar1(Row):ar1(Column), data = wheat)
```

```
> summary(wheat4.asr)$loglik
[1] -4220.261
```

```
> summary(wheat4.asr)$varcomp
```

	gamma	component	std.error	z.ratio	constraint
Variety	1.3482286	7.377148e+04	1.041705e+04	7.081800	Positive
units	0.5563897	3.044417e+04	8.074917e+03	3.770214	Positive
R!variance	1.0000000	5.471734e+04	1.062709e+04	5.148857	Positive
R!Row.ar1	0.8374999	8.374999e-01	4.486909e-02	18.665407	Unconstrained
R!Column.ar1	0.3753791	3.753791e-01	1.152641e-01	3.256687	Unconstrained

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("Variety:Column"=list("Column"=5.5)))
```

At present `asreml()` cannot average over `pol()` terms so we must specify the value of `Column` at which the predictions are to be formed; in this case we choose to form varietal predictions at the average value of `Column`, that is, 5.5.

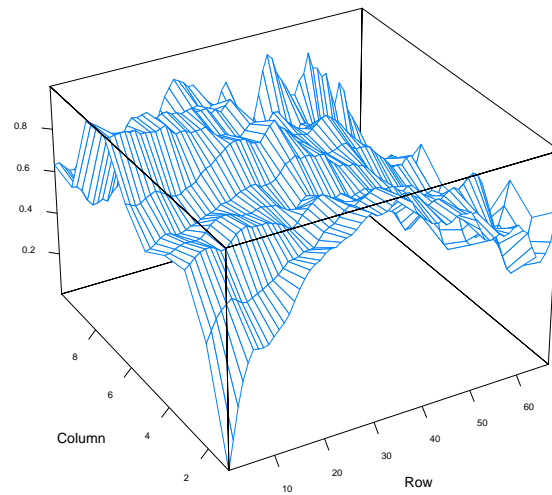


Figure 8.7: Sample variogram of the  $AR1 \times AR1 + \text{pol}(\text{column}, -1)$  model for the Tullibigeal data

```
> wheat4.pv$predictions
$"Variety:Column":
$"Variety:Column"$pvals:

      Column Variety predicted.value standard.error estStatus
1         5.5      1          2917.178         179.28821 Estimable
2         5.5      2          2957.741         178.76889 Estimable
3         5.5      3          2872.762         176.98813 Estimable
4         5.5      4          2986.473         178.74259 Estimable
...
522      5.5     522          2784.768         179.15427 Estimable
523      5.5     523          2904.942         179.53841 Estimable
524      5.5     524          2740.034         178.84664 Estimable
525      5.5     525          2669.956         179.24457 Estimable
526      5.5     526          2385.981           44.21617 Estimable
527      5.5     527          2697.068         133.44066 Estimable
528      5.5     528          2727.032         112.26513 Estimable
529      5.5     529          2699.824         103.90633 Estimable
530      5.5     530          3010.391         112.30814 Estimable
531      5.5     531          3020.072         112.25543 Estimable
532      5.5     532          3067.448         112.66467 Estimable

$"Variety:Column"$sved:
[1] 245.8018
```

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:Column"=list("Variety" = seq(1,525), "Column"=5.5)))

> wheat6.pv <- predict(wheat4.asr, classify="Variety:Column",
+ levels=list("Variety:Column"=list("Variety" = seq(526,532), "Column"=5.5)),
+ sed = list("Variety:Column"=T))
```

```
> wheat6.pv$predictions
$"Variety:Column":
$"Variety:Column"$pvals:
```

Notes:

```
- weed evaluated at average value of 0.459701
- pol(Column, -1) evaluated at 5.500000
- units terms are ignored unless specifically included
- mv is averaged over fixed levels
- pol(Column, -1) is included in the prediction
- weed is included in the prediction
- (Intercept) is included in the prediction
- units is ignored in this prediction
- mv is ignored in this prediction
```

	Column	Variety	predicted.value	standard.error	est.status
1	5.5	526	2385.981	44.21617	Estimable
2	5.5	527	2697.068	133.44066	Estimable
3	5.5	528	2727.032	112.26513	Estimable
4	5.5	529	2699.824	103.90633	Estimable
5	5.5	530	3010.391	112.30814	Estimable
6	5.5	531	3020.072	112.25543	Estimable
7	5.5	532	3067.448	112.66467	Estimable

```
"Variety:Column"$sed:
```

	[,1]	[,2]	[,3]	[,4]	[,5]	[,6]	[,7]
[1,]	NA	129.1900	107.2885	98.20983	107.4884	107.0353	107.4493
[2,]	129.18995	NA	165.5508	159.10100	165.5290	165.4920	165.9755
[3,]	107.28848	165.5508	NA	141.34763	148.5679	149.5352	148.2006
[4,]	98.20983	159.1010	141.3476	NA	143.0536	142.1024	143.3973
[5,]	107.48844	165.5290	148.5679	143.05365	NA	144.3514	149.5803
[6,]	107.03532	165.4920	149.5352	142.10244	144.3514	NA	149.5490
[7,]	107.44926	165.9755	148.2006	143.39734	149.5803	149.5490	NA

```
"Variety:Column"$avsed:
```

	min	mean	max
	98.20983	139.90452	165.97553

## 8.8 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.

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 subsections 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

```
> rice <- asreml.read.table("rice.txt", header=T)
> names(rice)
[1] "Pair" "rootwt" "Run" "sqrtroot" "Tmt" "Variety"
```

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).

The multivariate data frame

```
> riceMV <- asreml.read.table("riceMV.csv", header=T, sep=",")
> names(riceMV)
[1] "Pair" "Run" "Variety" "yc" "ye" "syc" "sye"
```

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).

A plot of the treated vs the control root area (on the square root scale) for each variety is shown in Figure 8.8. 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 8.8). 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

```
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 )
```

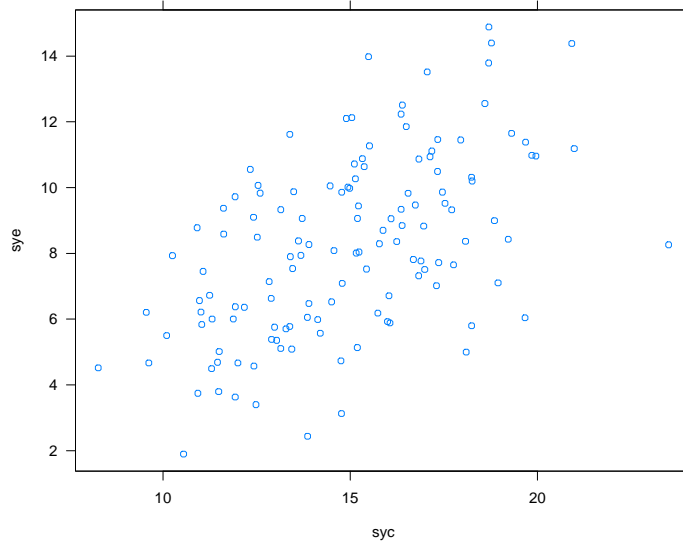


Figure 8.8: Rice bloodworm data: Plot of square root of root weight for treated versus control

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
```

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 `sqrtrwt` but rather two traits, namely `sqrtrwt` 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 `sqrtrwt` for the two groups (14.93 and 8.23 for control and treated respectively). The inclusion of `Tmt` as a fixed effect ensures that 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

$$\mathbf{y} = \mathbf{X}\boldsymbol{\tau} + \mathbf{Z}_1\mathbf{u}_1 + \mathbf{Z}_2\mathbf{u}_2 + \mathbf{Z}_3\mathbf{u}_3 + \mathbf{Z}_4\mathbf{u}_4 + \mathbf{Z}_5\mathbf{u}_5 + \mathbf{e} \quad (8.9)$$

where  $\mathbf{y}$  is a vector of length  $n = 264$  containing the `sqrtrwt` values,  $\boldsymbol{\tau}$  corresponds to a constant term and the fixed treatment contrast and  $\mathbf{u}_1 \dots \mathbf{u}_5$  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  $\text{var}(\mathbf{u}_i) = \sigma_i^2 \mathbf{I}_{b_i}$  (where  $b_i$  is the length of  $\mathbf{u}_i$ ,  $i = 1 \dots 5$ ) and  $\text{var}(\mathbf{e}) = \sigma^2 \mathbf{I}_n$ .

The `asreml()` call is:

```

> rice1.asr <- asreml(sqrtroot ~ Tmt, random = ~ Variety+Variety:Tmt+Run+Pair+Run:Tmt,
+ data = rice)

> summary(rice1.asr)$loglik
[1] -345.2559

> summary(rice1.asr)$varcomp

           gamma component std.error  z.ratio constraint
Variety    1.8085661 2.3782170 0.7914703 3.004809  Positive
Variety:Tmt 0.3743885 0.4923110 0.2764182 1.781037  Positive
Run        0.2444393 0.3214312 0.5482284 0.586309  Positive
Pair       0.7421389 0.9758932 0.3883409 2.512981  Positive
Tmt:Run    1.3291572 1.7478068 0.4793480 3.646217  Positive
R!variance 1.0000000 1.3149738 0.2974417 4.420946  Positive

> 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  1953.17      1485.33 < 2.2e-16 ***
Tmt          1   617.16      469.33 < 2.2e-16 ***
residual (MS)          1.31

```

The estimated variance components from this analysis also appear in column (a) of Table 8.8. 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  $\text{var}(\mathbf{u}_2) = \sigma_2^2 \mathbf{I}_{88}$  we assume

$$\text{var}(\mathbf{u}_2) = \begin{bmatrix} \sigma_{2c}^2 & 0 \\ 0 & \sigma_{2t}^2 \end{bmatrix} \otimes \mathbf{I}_{44}$$

where  $\sigma_{2c}^2$  and  $\sigma_{2t}^2$  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()` sets the parameter constraint for variance components to **Positive**. To allow for negative components, which may have meaning in this particular example, we must set the parameter constraints to **Unconstrained**. The following sequence of calls

- creates default **R** and **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 **G** level parameter settings in `RG.rice` through the `G.param` argument.

```

> temp <- asreml(sqrtroot ~ Tmt,
+ random = ~ Variety+Variety:diag(Tmt)+Run+Pair+ Run:diag(Tmt)+at(Tmt,2):units,
+ data = rice, start.values="gammascsv")

```

```
# set variance constraint codes to U

> rice2.asr <- asreml(sqrtroot ~ Tmt,
+ random = ~ Variety+Variety:diag(Tmt)+Run+Pair+ Run:diag(Tmt)+at(Tmt,2):units,
+ G.param = "gammas.csv, data = rice)

> summary(rice2.asr)$loglik
[1] -343.2199
> summary(worm2.asr)$varcomp

              gamma component std.error z.ratio  constraint
Variety              2.018113  2.333895  0.776098  3.007    Positive
Variety:Tmt!Tmt.Control.var  1.302060  1.505799  0.665177  2.263 Unconstrained
Variety:Tmt!Tmt.Exposed.var -0.321861 -0.372224  0.456151 -0.816 Unconstrained
Run                   0.276148  0.319358  0.543446  0.587    Positive
Pair                  0.853857  0.987463  0.381194  2.590    Positive
Tmt:Run!Tmt.Control.var   1.200871  1.388777  0.635834  2.184 Unconstrained
Tmt:Run!Tmt.Exposed.var   1.923409  2.224372  0.723924  3.072 Unconstrained
at(Tmt, Exposed):units    0.176145  0.203707  0.631838  0.322    Positive
R!variance                1.000000  1.156474  0.417382  2.770    Positive

> 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  1476.86      1277.03 < 2.2e-16 ***
Tmt          1   519.01       448.78 < 2.2e-16 ***
residual (MS)      1.16
```

The estimated variance components from this analysis are given in column (b) of Table 8.8. 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,

$$\text{var}(\mathbf{1}_2 \otimes \mathbf{u}_1 + \mathbf{u}_2) = \begin{bmatrix} \sigma_1^2 + \sigma_{2c}^2 & \sigma_1^2 \\ \sigma_1^2 & \sigma_1^2 + \sigma_{2t}^2 \end{bmatrix} \otimes \mathbf{I}_{44} \quad (8.10)$$

Using the estimates from Table 8.8 this structure is estimated as

$$\begin{bmatrix} 3.84 & 2.33 \\ 2.33 & 1.96 \end{bmatrix} \otimes \mathbf{I}_{44}$$

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 8.8).

Table 8.8: 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

source	(a)	(b)	
	homogeneous	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	

### 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

$$\mathbf{y}_j = \mathbf{X}\boldsymbol{\tau}_j + \mathbf{Z}_v\mathbf{u}_{v_j} + \mathbf{Z}_r\mathbf{u}_{r_j} + \mathbf{e}_j \quad (j = c, t) \quad (8.11)$$

where  $\mathbf{y}_j$  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  $\mathbf{u}_{v_j}$  and  $\mathbf{u}_{r_j}$  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  $\text{var}(\mathbf{u}_{v_j}) = \sigma_{v_j}^2 \mathbf{I}_{44}$ ,  $\text{var}(\mathbf{u}_{r_j}) = \sigma_{r_j}^2 \mathbf{I}_{66}$  and  $\text{var}(\mathbf{e}_j) = \sigma_j^2 \mathbf{I}_{132}$ . The bivariate model can be written as a direct extension of (8.11), namely

$$\mathbf{y} = (\mathbf{I}_2 \otimes \mathbf{X}) \boldsymbol{\tau} + (\mathbf{I}_2 \otimes \mathbf{Z}_v) \mathbf{u}_v + (\mathbf{I}_2 \otimes \mathbf{Z}_r) \mathbf{u}_r + \mathbf{e}^* \quad (8.12)$$

where  $\mathbf{y} = (\mathbf{y}'_c, \mathbf{y}'_t)'$ ,  $\mathbf{u}_v = (\mathbf{u}'_{v_c}, \mathbf{u}'_{v_t})'$ ,  $\mathbf{u}_r = (\mathbf{u}'_{r_c}, \mathbf{u}'_{r_t})'$  and  $\mathbf{e}^* = (\mathbf{e}'_c, \mathbf{e}'_t)'$ .

There is an equivalence between the effects in this bivariate model and the univariate model of (8.9). The variety effects for each trait ( $\mathbf{u}_v$  in the bivariate model) are partitioned in (8.9) into variety main effects and **tmt.variety** interactions so that  $\mathbf{u}_v = \mathbf{1}_2 \otimes \mathbf{u}_1 + \mathbf{u}_2$ . There is a similar partitioning for the run effects and the errors (Table 8.9).

In addition to the assumptions in the models for individual traits (8.11), the bivariate analysis involves the assumptions  $\text{cov}(\mathbf{u}_{v_c}) \mathbf{u}'_{v_t} = \sigma_{v_{ct}} \mathbf{I}_{44}$ ,  $\text{cov}(\mathbf{u}_{r_c}) \mathbf{u}'_{r_t} = \sigma_{r_{ct}} \mathbf{I}_{66}$  and  $\text{cov}(\mathbf{e}_c) \mathbf{e}'_t = \sigma_{ct} \mathbf{I}_{132}$ . 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

$$\text{var}(\mathbf{u}_v) = \begin{bmatrix} \sigma_{v_c}^2 & \sigma_{v_{ct}} \\ \sigma_{v_{ct}} & \sigma_{v_t}^2 \end{bmatrix} \otimes \mathbf{I}_{44}$$

This unstructured form for **trait:Variety** in the bivariate analysis is equivalent to the **Variety** main effect plus heterogeneous **Variety:Tmt** interaction variance structure (8.10) in the univariate analysis. Similarly the unstructured form for **trait:Run** is equivalent to the **Run** main effect

Table 8.9: Equivalence of random effects in bivariate and univariate analyses

effects	bivariate (model 8.12)	univariate (model 8.9)
trait:Variety	$\mathbf{u}_v$	$\mathbf{1}_2 \otimes \mathbf{u}_1 + \mathbf{u}_2$
trait:Run	$\mathbf{u}_r$	$\mathbf{1}_2 \otimes \mathbf{u}_3 + \mathbf{u}_4$
Pair:trait	$\mathbf{e}^*$	$\mathbf{1}_2 \otimes \mathbf{u}_5 + \mathbf{e}$

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() call is:

```
> riceMV.asr <- asreml(cbind(syc,sye) ~ trait, + random = ~ us(trait):Variety + us(trait):Run,
+ rcov = ~ units:us(trait), data = riceMV)
```

```
> summary(riceMV.asr)$loglik
[1] -343.2199
```

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

	gamma	component	std.error	z.ratio	constraint
trait:Variety!trait.syc:syc	3.8386404	3.8386404	1.1059311	3.4709	Unconstrained
trait:Variety!trait.sye:syc	2.3332757	2.3332757	0.7755975	3.0083	Unconstrained
trait:Variety!trait.sye:sye	1.9612537	1.9612537	0.7281807	2.6933	Unconstrained
trait:Run!trait.syc:syc	1.7083269	1.7083269	0.6534826	2.6141	Unconstrained
trait:Run!trait.sye:syc	0.3196590	0.3196590	0.5437117	0.5879	Unconstrained
trait:Run!trait.sye:sye	2.5436703	2.5436703	0.7957811	3.1964	Unconstrained
R!variance	1.0000000	1.0000000	NA	NA	Fixed
R!trait.syc:syc	2.1436774	2.1436774	0.4822556	4.4451	Unconstrained
R!trait.sye:syc	0.9873042	0.9873042	0.3811844	2.5900	Unconstrained
R!trait.sye:sye	2.3474190	2.3474190	0.5076600	4.6239	Unconstrained

The resultant REML log-likelihood is identical to that of the heterogeneous univariate analysis (column (b) of Table 8.8). The estimated variance parameters are summarised in Table 8.10.

Table 8.10: 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

Predicted variety means are obtained from:

```

> riceMV.pv <- predict(riceMV.asr, classify="trait:Variety")

> riceMV.pv$predictions
$"trait:Variety":
$"trait:Variety"$pvals:

   trait      Variety predicted.value standard.error est.status
1   syc    AliCombo    14.953229      0.9180964  Estimable
2   syc     Amaroo    16.161198      0.9180985  Estimable
3   syc    Balilla    14.420236      0.9185701  Estimable
4   syc  Bluebelle    13.103132      0.9309747  Estimable
5   syc     Bogan    15.768223      0.9548522  Estimable
...
84  sye      TKM6      9.807568      0.8056834  Estimable
85  sye    WC1403      9.287950      0.8057545  Estimable
86  sye  WC140311      8.923817      0.8056858  Estimable
87  sye      YRK1      8.335681      0.8190248  Estimable
88  sye      YRK3      8.113448      0.8190248  Estimable

$"trait:Variety"$sveds:
[1] 1.214741

```

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  $\text{var}(y) = (dy/dx)_\theta^2 \text{var}(x)$ . 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$predictions$"trait:Variety"$pvals
> pv$rootwt <- pv$predicted.value^2
> pv$approxSE <- sqrt(4*pv$predicted.value^2 * pv$standard.error^2)
> pv$est.status <- NULL

> pv
   trait      Variety predicted.value standard.error  rootwt approxSE
1   syc    AliCombo    14.953229      0.9180964  223.5991  27.45701
2   syc     Amaroo    16.161198      0.9180985  261.1843  29.67514
3   syc    Balilla    14.420236      0.9185701  207.9432  26.49199
4   syc  Bluebelle    13.103132      0.9309747  171.6921  24.39737
5   syc     Bogan    15.768223      0.9548522  248.6368  30.11265
...
84  sye      TKM6      9.807568      0.8056834   96.1884  15.80359
85  sye    WC1403      9.287950      0.8057545   86.2660  14.96762
86  sye  WC140311      8.923817      0.8056858   79.6345  14.37959
87  sye      YRK1      8.335681      0.8190248   69.4836  13.65426
88  sye      YRK3      8.113448      0.8190248   65.8280  13.29023

```

### 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

$$\mathbf{u}_{v_t} = \beta \mathbf{u}_{v_c} + \boldsymbol{\epsilon}$$

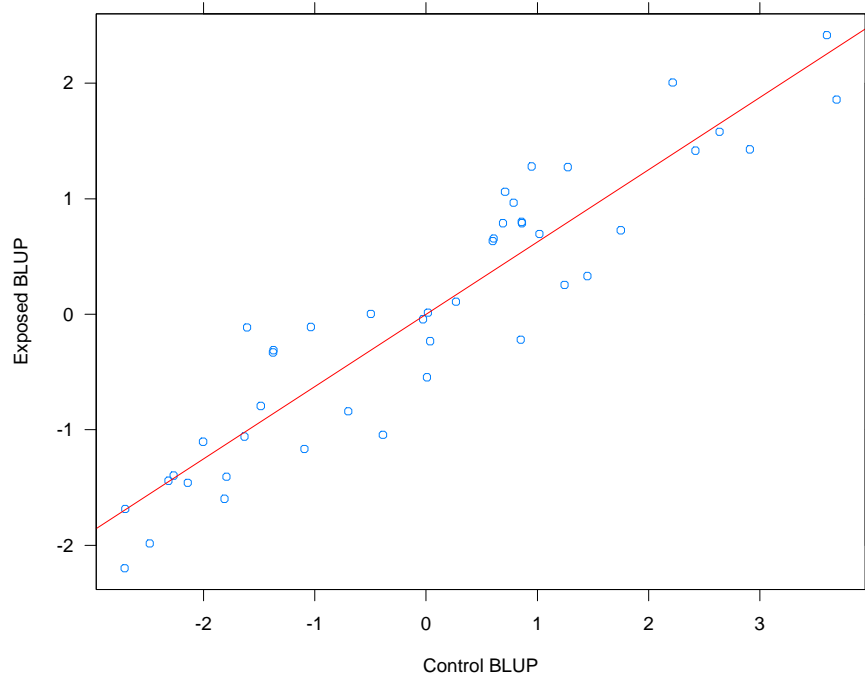


Figure 8.9: BLUPs for treated plotted against BLUPs for control

where the slope  $\beta = \sigma_{v_{ct}} / \sigma_{v_c}^2$ .

Tolerance can be defined in terms of the deviations from regression,  $\epsilon$ . 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 8.9 the BLUPs for treated have been plotted against the 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.

An alternative definition of tolerance is the simple difference between treated and control BLUPs for each variety, namely  $\delta = \mathbf{u}_{v_c} - \mathbf{u}_{v_t}$ . Unless  $\beta = 1$  the two measures  $\epsilon$  and  $\delta$  have very different interpretations. The key difference is that  $\epsilon$  is a measure which is *independent* of inherent vigour whereas  $\delta$  is not. To see this consider

$$\begin{aligned} \text{cov}(\epsilon) \mathbf{u}'_{v_c} &= \text{cov}(\mathbf{u}_{v_t} - \beta \mathbf{u}_{v_c}) \mathbf{u}'_{v_c} \\ &= \left( \sigma_{v_{ct}} - \frac{\sigma_{v_{ct}}}{\sigma_{v_c}^2} \sigma_{v_c}^2 \right) \mathbf{I}_{44} \\ &= \mathbf{0} \end{aligned}$$

whereas

The independence of  $\epsilon$  and  $\mathbf{u}_{v_c}$  and dependence between  $\delta$  and  $\mathbf{u}_{v_c}$  is clearly illustrated in Figures 8.10 and 8.11. 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.

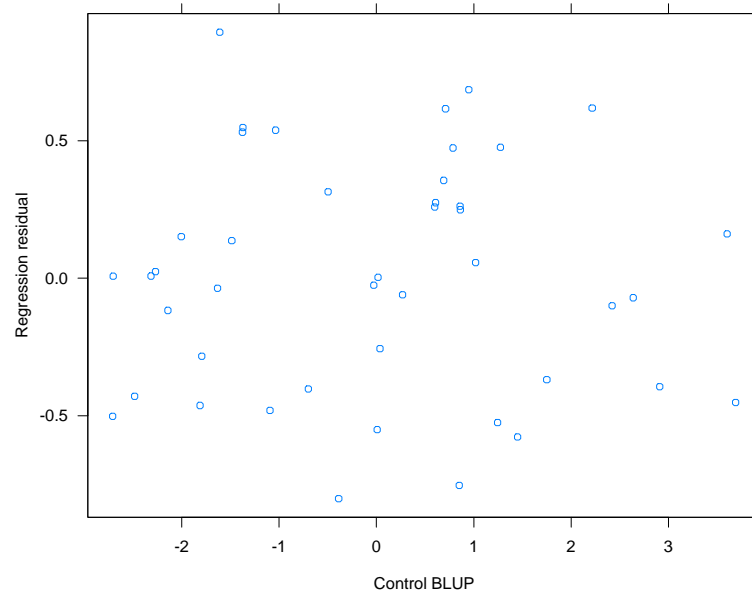


Figure 8.10: Estimated deviations from regression of treated on control for each variety plotted against estimate for control

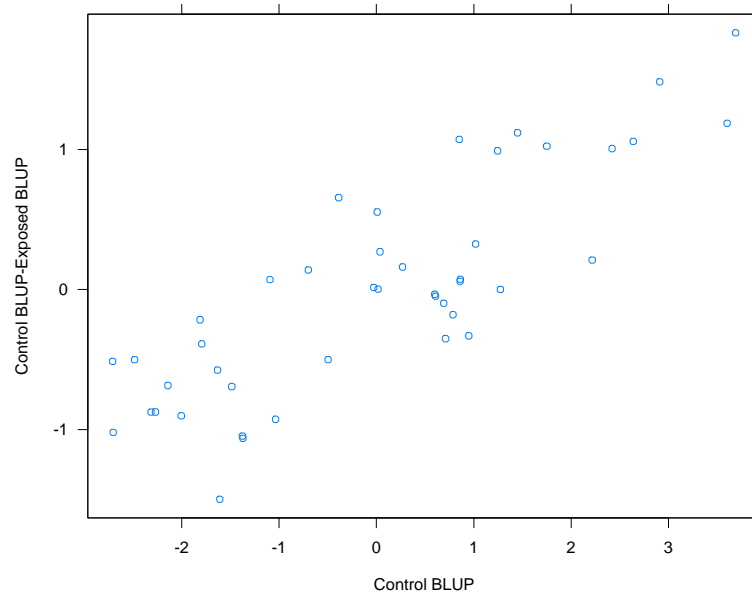


Figure 8.11: Estimated difference between control and treated for each variety plotted against estimate for control

## 8.9 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()` 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 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 reanalysed by Pinheiro and Bates [2000, p338]. The data are trunk circumferences (in millimetres) of each of 5 trees taken at 7 times (Figure 8.12). 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:

$$y = \frac{\phi_1}{1 + \exp[-(x - \phi_2)/\phi_3]} \quad (8.13)$$

where  $y$  is the trunk circumference,  $x$  is the tree age in days since December 31 1968,  $\phi_1$  is the asymptotic height,  $\phi_2$  is the inflection point or the time at which the tree reaches  $0.5\phi_1$ ,  $\phi_3$  is the time elapsed between trees reaching half and about  $3/4$  of  $\phi_1$ .

The data frame `orange` contains:

```
> orange <- asreml.read.table("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`).

Initially we restrict the dataset to tree 1 to demonstrate fitting cubic splines in `asreml()`. 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,  $\delta$  whose elements  $\delta_i, i = 2, \dots, 6$  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].

```
> orange.asr <- asreml(circ ~ x, random = ~ spl(x),
+ splinepoints = list(x = c(118,484,664,1004,1231,1372,1582)),
+ data = orange, subset = Tree==1)
```

In this example the spline knot points are specifically given in the `splinepoints` 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 `splinepoints` argument was omitted.

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

      gamma component std.error  z.ratio constraint
spl(x)    0.07876884  3.954159  9.950608  0.3973786  Positive
R!variance 1.00000000 50.199529 37.886791 1.3249876  Positive
```

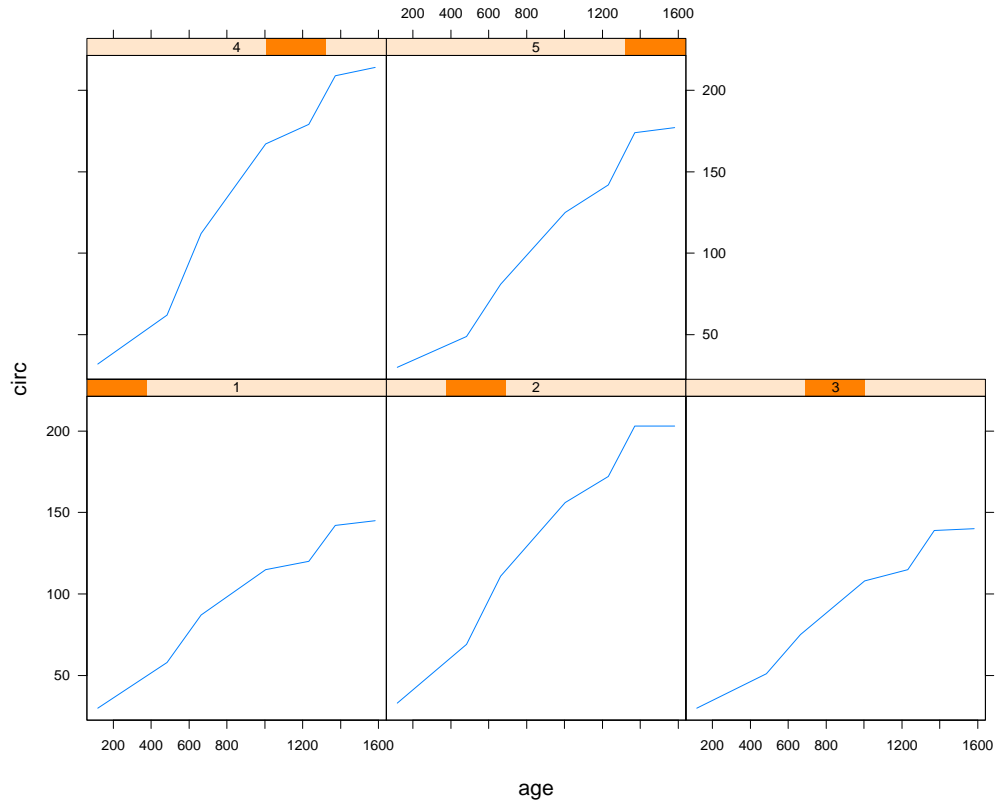


Figure 8.12: Trellis plot of trunk circumference (mm) for each tree against age in days since 1 December 1968.

```
> wald(orange.asr, denDF="default")

$Wald
      Df denDF  F inc      Pr
(Intercept)  1   3.5 1382.0 3.124431e-06
x             1   3.5  217.5 1.229875e-04

$stratumVariances
      df Variance  spl(x) R!variance
spl(x)  1.488944 97.64263 11.99606      1
R!variance 3.511056 50.20223  0.00000      1
```

Predicted values of the spline curve at nominated points can be obtained by:

```
> orange.pv <- predict(orange.asr, classify = "x", predictpoints=list(x=seq(150,1500,50)))
```

The `predictpoints` argument adds the nominated points to the design matrix for prediction purposes (Figure 8.13). Note that `predictpoints` could have been included in the `asrem1()` call instead of `predict` and if omitted, 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 8.13) 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 8.11, following Verbyla et al. [1999].

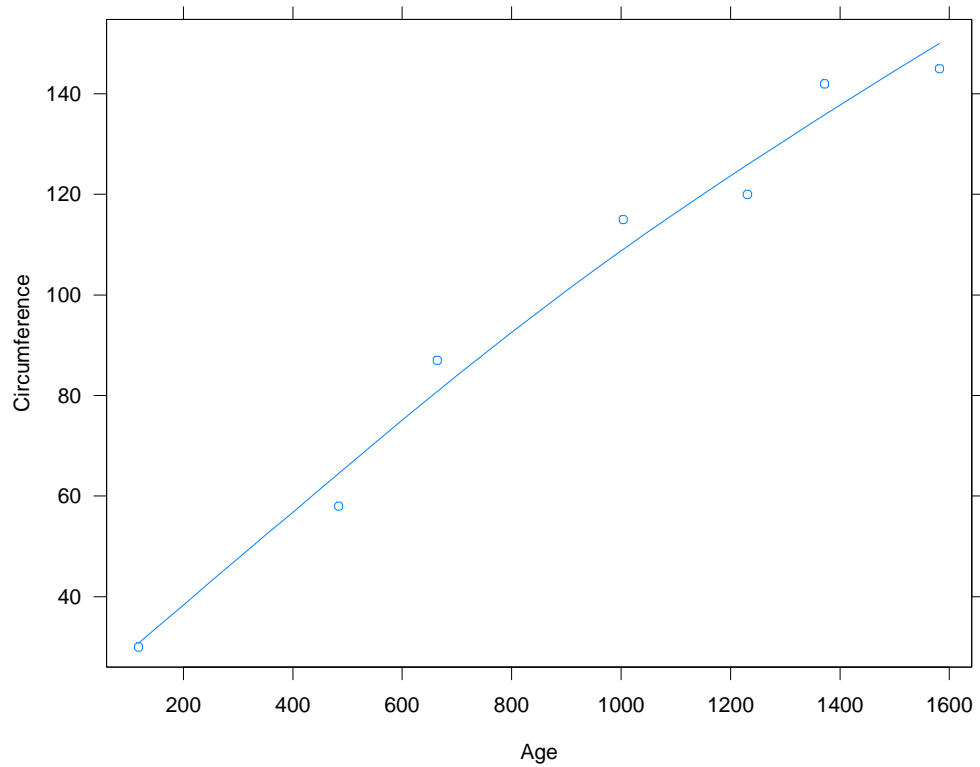


Figure 8.13: Fitted cubic smoothing spline for tree 1

Table 8.11: 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  $\mathbf{rc}$  in Table 8.11). Thus, if  $\mathbf{U}^{5 \times 2}$  is the matrix of intercepts (column 1) and slopes (column 2) for each tree, then we assume that

$$\text{var}(\text{vec}(\mathbf{U})) = \mathbf{\Sigma} \otimes \mathbf{I}_5$$

where  $\mathbf{\Sigma}$  is a  $2 \times 2$  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  $\text{dev}(\mathbf{x})$  as a random term. The full model is:

```
> orange1.asr <- asreml(circ ~ x,
+ random = ~ str(~ Tree/x, ~ diag(2):id(5)) + spl(x)+spl(x):Tree+dev(x),
+ splinepoints = list(x = c(118,484,664,1004,1231,1372,1582)), data = orange)
```

Table 8.12 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,  $\text{dev}(\mathbf{x})$ ).

Table 8.12: Sequence of models fitted to the orange data

term	model					
	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

The `asreml()` call and variance components for model 1 are:

```
> orange1.asr <- asreml(circ ~ x, random = ~ str(~ Tree/x, ~ diag(2):id(5)) + spl(x) + spl(x):Tree,
+ splinepoints = list(x = c(118,484,664,1004,1231,1372,1582)), data = orange)
```

```
> summary(orange1.asr)$varcomp
```

	gamma	component	std.error	z.ratio	constraint
Tree+Tree:x!diag(2).1.var	4.789034e+00	3.044970e+01	2.457813e+01	1.238894	Positive
Tree+Tree:x!diag(2).2.var	9.392257e-05	5.971797e-04	4.240625e-04	1.408235	Positive
spl(x)	1.004896e+02	6.389346e+02	4.131293e+02	1.546573	Positive
Tree:spl(x)	1.116746e+00	7.100507e+00	4.935166e+00	1.438757	Positive
R!variance	1.000000e+00	6.358213e+00	3.652341e+00	1.740860	Positive

The fitted curves from this model are shown in Figure 8.14. The fit is unacceptable because the spline has picked up too much curvature, suggesting there may be systematic non-smooth variation

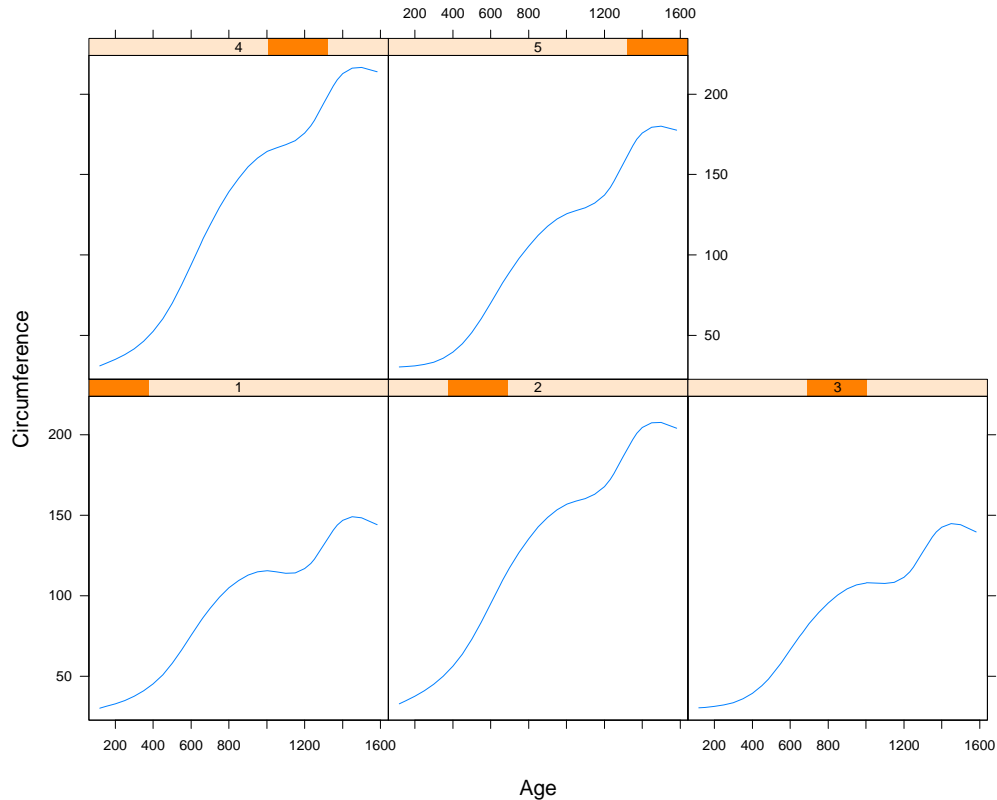


Figure 8.14: Plot of fitted cubic smoothing spline for model 1

at the overall level. This can be formally examined by including the  $\text{dev}(\mathbf{x})$  term as a random effect.

Model 2 increased the REML log-likelihood by 3.70 ( $P < 0.05$ ) with the  $\text{spl}(\mathbf{x})$  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  $\text{dev}(\mathbf{x})$  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()` 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, ~ us(2,init=c(5.0,-0.01,0.0001)):id(5)) + spl(x) + spl(x):Tree,
splinepoints = list(x = c(118,484,664,1004,1231,1372,1582)), data = orange)

> summary(orange6.asr)$varcomp
```

	gamma	component	std.error	z.ratio	constraint
Tree+Tree:x!us(2).1:1	5.6011527956	31.7916580507	2.512945e+01	1.2651157	Unconstrained
Tree+Tree:x!us(2).2:1	-0.0123766915	-0.0702490288	8.234755e-02	-0.8530798	Unconstrained
Tree+Tree:x!us(2).2:2	0.0001080119	0.0006130664	4.344414e-04	1.4111604	Unconstrained
spl(x)	2.1666162231	12.2975259925	1.119901e+01	1.0980902	Positive
Tree:spl(x)	1.3751301469	7.8051195890	5.222912e+00	1.4944001	Positive
R!variance	1.0000000000	5.6759133719	3.294276e+00	1.7229622	Positive

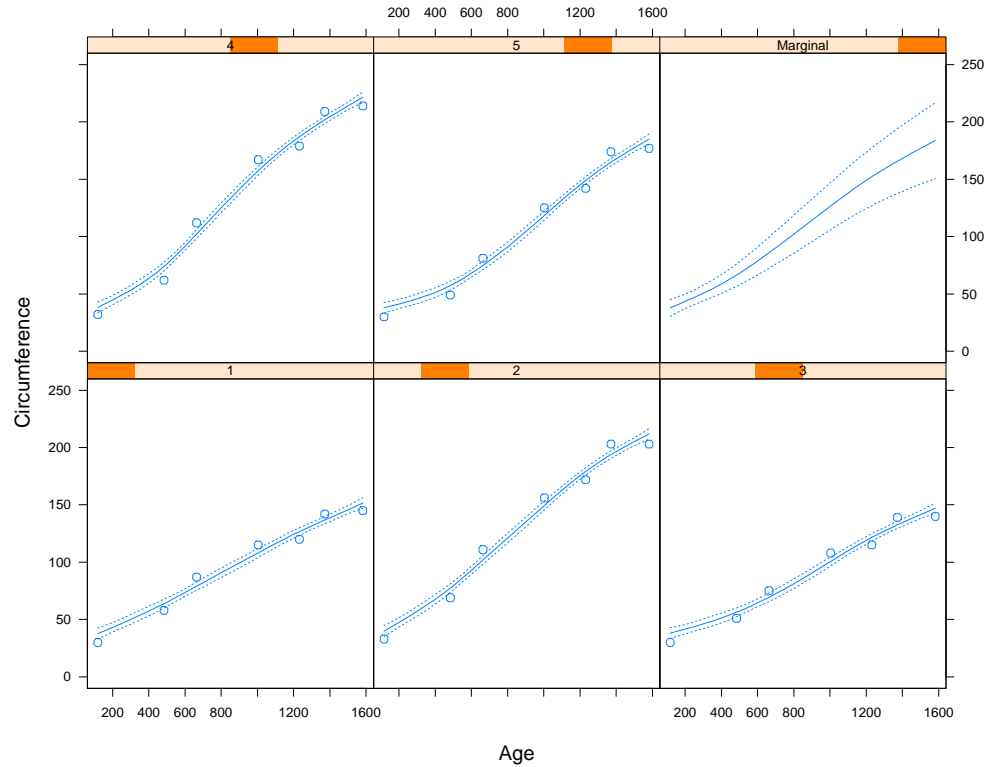


Figure 8.15: Fitted values adjusted for **Season** and approximate confidence intervals for model 6

The fitted values for individual trees (adjusted for **Season**) from model 6 together with a marginal prediction and approximate confidence intervals ( $2 \times$  standard error of prediction) are shown in Figure 8.15. 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 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.

# Some technical details about model fitting in `asreml()`

A

## 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()`

Solutions for the fixed and random effects in linear mixed model analysis using `asreml()` 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

[@thispage /FitH @ypos]::

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 = $y \sim -1 + \text{Var}$ , random = $\sim \text{Rep}$	0	<b>Var</b> fully fitted
fixed = $y \sim \text{Var}$ , random = $\sim \text{Rep}$	1	first level of <b>Var</b> dropped
fixed = $y \sim -1 + \text{Var} + \text{Trt}$ , random = $\sim \text{Rep}$	1	<b>Var</b> fully specified, first level of <b>Trt</b> dropped from the models
fixed = $y \sim \text{Var} + \text{Trt} + \text{Var:Trt}$ , random = $\sim \text{Rep}$	8	first level of both <b>Var</b> and <b>Trt</b> dropped from the model, together with subsequent interactions
fixed = $\sim \text{Var} + \text{Trt}$ , random = $\sim \text{Rep}$ , sparse = $\sim \text{Var:Trt}$	8	<b>Var:Trt</b> fully specified; (Intercept), <b>Var</b> and <b>Trt</b> completely singular and dropped from the model

# Available variance models

## B

Table B.1 presents the full range of variance models available in `asreml()` 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  $\mathbf{C}^{(\omega \times \omega)} = [C_{ij}]$  be the correlation matrix for a particular correlation model. If  $\mathbf{\Sigma}^{(\omega \times \omega)}$  is the corresponding homogeneous variance matrix then

$$\mathbf{\Sigma} = \sigma^2 \mathbf{C}$$

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  $\mathbf{\Sigma} = \sigma^2 I_\omega$  (specified `idv()` in the `asreml()` function call, see below) and one parameter. Likewise, if  $\mathbf{\Sigma}_h^{(\omega \times \omega)}$  is the heterogeneous variance matrix corresponding to  $\mathbf{C}$ , then

$$\mathbf{\Sigma}_h = \mathbf{D} \mathbf{C} \mathbf{D}$$

where  $\mathbf{D}^{(\omega \times \omega)} = \text{diag}(\sigma_i)$ . In this case there are an additional  $\omega$  parameters. For example, the `asreml()` function for the heterogeneous variance model corresponding to `id()` variance model has variance matrix

$$\mathbf{\Sigma}_h = \begin{bmatrix} \sigma_1^2 & 0 & \dots & 0 \\ 0 & \sigma_2^2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \sigma_\omega^2 \end{bmatrix}$$

(specified `idh()` in the `asreml()` command file, see below) and involves the  $\omega$  parameters  $\sigma_1^2 \dots \sigma_\omega^2$ .

Table B.1: Details of the available variance models

function name	description	algebraic form	number of parameters		
			corr	hom variance	het variance
<b>Correlation models</b>					
id()	identity	$C_{ii} = 1, C_{ij} = 0, i \neq j$	0	1	$\omega$
ar1()	1 <sup>st</sup> order autoregressive	$C_{ii} = 1, C_{i+1,i} = \phi_1$ $C_{ij} = \phi_1 C_{i-1,j}, i > j + 1$ $ \phi_1  < 1$	1	2	$1 + \omega$
ar2()	2 <sup>nd</sup> order autoregressive	$C_{ii} = 1,$ $C_{i+1,i} = \phi_1 / (1 - \phi_2)$ $C_{ij} = \phi_1 C_{i-1,j} + \phi_2 C_{i-2,j}, i > j + 1$ $ \phi_1 \pm \phi_2  < 1,$ $ \phi_1  < 1,  \phi_2  < 1$	2	3	$2 + \omega$
ar3()	3 <sup>rd</sup> order autoregressive	$C_{ii} = 1, \Omega = 1 - \phi_2 - \phi_3(\phi_1 + \phi_3),$ $C_{i+1,i} = (\phi_1 + \phi_2\phi_3)/\Omega,$ $C_{i+2,i} = (\phi_1(\phi_1 + \phi_3) + \phi_2(1 - \phi_2))/\Omega,$ $C_{ij} = \phi_1 C_{i-1,j} + \phi_2 C_{i-2,j} + \phi_3 C_{i-3,j}, i > j + 2$ $ \phi_1  < (1 - \phi_2),  \phi_2  < 1,  \phi_3  < 1$	3	4	$3 + \omega$
sar()	symmetric autoregressive	$C_{ii} = 1,$ $C_{i+1,i} = \phi_1 / (1 + \phi_1^2/4)$ $C_{ij} = \phi_1 C_{i-1,j} - \phi_1^2/4 C_{i-2,j},$ $i > j + 1$ $ \phi_1  < 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 <sup>st</sup> order moving average	$C_{ii} = 1,$ $C_{i+1,i} = -\theta_1 / (1 + \theta_1^2)$ $C_{ji} = 0, j > i + 2$ $ \theta_1  < 1$	1	2	$1 + \omega$
ma2()	2 <sup>nd</sup> order moving average	$C_{ii} = 1,$ $C_{i+1,i} = -\theta_1(1 - \theta_2) / (1 + \theta_1^2 + \theta_2^2)$ $C_{i+2,i} = -\theta_2 / (1 + \theta_1^2 + \theta_2^2)$ $C_{ji} = 0, j > i + 2$ $\theta_2 \pm \theta_1 < 1$ $ \theta_1  < 1,  \theta_2  < 1$	2	3	$2 + \omega$

Details of the available variance models

function name	description	algebraic form	number of parameters		
			corr	hom variance	het variance
arma()	autoregressive moving average	$C_{ii} = 1,$ $C_{i+1,i} = (\theta - \phi)(1 - \theta\phi)/(1 + \theta^2 - 2\theta\phi)$ $C_{ji} = \phi C_{j-1,i}, j > i + 1$ $ \theta  < 1,  \phi  < 1$	2	3	$2 + \omega$
cor()	uniform correlation	$C_{ii} = 1, C_{ij} = \theta, i \neq j$	1	2	$1 + \omega$
corb()	banded correlation	$C_{ii} = 1$ $C_{i+j,i} = \phi_j, 1 \leq j \leq \omega - 1$ $ \phi_j  < 1$	$\omega - 1$	$\omega$	$2\omega - 1$
corg()	general correlation corgh() = us()	$C_{ii} = 1$ $C_{ij} = \phi_{ij}, i \neq j$ $ \phi_{ij}  < 1$	$\frac{\omega(\omega-1)}{2}$	$\frac{\omega(\omega-1)}{2} + 1$	$\frac{\omega(\omega-1)}{2} + \omega$
<b>One-dimensional equally spaced power models</b>					
exp()	exponential	$C_{ii} = 1$ $C_{ij} = \phi^{ x_i - x_j }, i \neq j$ $x_i$ are <i>coordinates</i> $ \phi  < 1$	1	2	$1 + \omega$
gau()	gaussian	$C_{ii} = 1$ $C_{ij} = \phi^{(x_i - x_j)^2}$ $x_i$ are <i>coordinates</i> $ \phi  < 1$	1	2	$1 + \omega$
<b>Two-dimensional irregularly spaced power models</b>					
iexp()	isotropic exponential	$C_{ii} = 1$ $C_{ij} = \phi^{ x_i - x_j  +  y_i - y_j }$ $\mathbf{x}$ and $\mathbf{y}$ <i>vectors of coordinates</i> $ \phi  < 1$	1	2	$1 + \omega$
igau()	isotropic gaussian	$C_{ii} = 1$ $C_{ij} = \phi^{(x_i - x_j)^2 + (y_i - y_j)^2}$ $\mathbf{x}$ and $\mathbf{y}$ <i>vectors of coordinates</i> $ \phi  < 1$	1	2	$1 + \omega$
ieuc()	isotropic euclidean	$C_{ii} = 1$ $C_{ij} = \phi^{\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}}$ $\mathbf{x}$ and $\mathbf{y}$ <i>vectors of coordinates</i> $ \phi  < 1$	1	2	$1 + \omega$

Details of the available variance models

function name	description	algebraic form	number of parameters		
			corr	hom variance	het variance
isp()	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])	$C_{ij} = 1 - \frac{2}{\pi}(\theta_{ij}\sqrt{1 - \theta_{ij}^2} + \sin^{-1}\theta_{ij})$ $0 < \phi_1$	1	2	$1 + \omega$
aexp()	anisotropic exponential	$C_{ii} = 1$ $C_{ij} = \phi_1^{ x_i - x_j } \phi_2^{ y_i - y_j }$ <b>x</b> and <b>y</b> vectors of coordinates $ \phi_1  < 1,  \phi_2  < 1$	2	3	$2 + \omega$
agau()	anisotropic gaussian	$C_{ii} = 1$ $C_{ij} = \phi_1^{(x_i - x_j)^2} \phi_2^{(y_i - y_j)^2}$ <b>x</b> and <b>y</b> vectors of coordinates $ \phi_1  < 1,  \phi_2  < 1$	2	3	$2 + \omega$
mtrn()	Matérn with first $1 \leq k \leq 5$ parameters specified by the user	$C_{ij}$ = Matérn: see Section 4.3.3 $\phi > 0$ range, $\nu$ shape(0.5) $\delta > 0$ anisotropy ratio(1), $\alpha$ anisotropy angle(0), $\lambda(1 2)$ metric(2)	$k$	$k+1$	$k + \omega$
<b>Heterogeneous variance models</b>					
diag()	diagonal = idh()	$\Sigma_{ii} = \phi_i \Sigma_{ij} = 0, i \neq j$	-	-	$\omega$
us()	unstructured general covariance matrix	$\Sigma_{ij} = \phi_{ij}$	-	-	$\frac{\omega(\omega+1)}{2}$
ante( $k$ )	antependence order $k$ $1 \leq \text{order} \leq \omega - 1$	$\Sigma^{-1} = \mathbf{U}\mathbf{D}\mathbf{U}'$ $D_{ii} = d_i, D_{ij} = 0, i \neq j$ $U_{ii} = 1, U_{ij} = u_{ij}, 1 \leq j - i \leq \text{order}$ $U_{ij} = 0, i > j$	-	-	$(k + 1)(\omega - k/2)$
chol( $k$ )	cholesky order $k$ $1 \leq \text{order} \leq \omega - 1$	$\Sigma = \mathbf{L}\mathbf{D}\mathbf{L}'$ $D_{ii} = d_i, D_{ij} = 0, i \neq j$ $L_{ii} = 1, L_{ij} = l_{ij}, 1 \leq i - j \leq \text{order}$	-	-	$(k + 1)(\omega - k/2)$
fa( $k$ )	factor analytic order $k$	$\Sigma = \mathbf{\Gamma}\mathbf{\Gamma}' + \mathbf{\Psi}$ , $\mathbf{\Gamma}$ contains covariance factors $\mathbf{\Psi}$ contains specific variance	-	-	$k\omega + \omega$

## Details of the available variance models

function name	description	algebraic form	number of parameters		
			corr	hom variance	het variance
<b>Inverse relationship matrices</b>					
giv()	generalised inverse		-	1	-
ped()	inverse relationship matrix derived from pedigree		-	1	-
<b>General variance models</b>					
str()	variance model relating to a sequence of terms in the model		-	1	-

# Bibliography

- G. E. P. Box. Analysis of growth and wear curves. *Biometrics*, 6:362–389, 1950.
- N. E. Breslow and D. G. Clayton. Approximate inference in generalized linear mixed models. *Journal of the American Statistical Association*, 88:9–25, 1993.
- N.E. Breslow. Whither PQL? Technical Report 192, UW Biostatistics Working Paper Series, University of Washington, 2003. URL <http://www.bepress.com/uwbiostat/paper192/>.
- N.E. Breslow and X. Lin. Bias correction in generalised linear mixed models with a single component of dispersion. *Biometrika*, 82:81–91, 1995.
- D. R. Cox and D. V. Hinkley. *Theoretical Statistics*. London: Chapman and Hall, 1974.
- D. R. Cox and E. J. Snell. *Applied Statistics; Principles and Examples*. London: Chapman and Hall, 1981.
- N. A. C. Cressie. *Statistics for spatial data*. New York: John Wiley and Sons, Inc., 1991.
- B. R. Cullis, A. B. Frensham, and F. M. Thomson. Efficient variety trialling systems in the Southern region. Grains Research and Development Corporation Final Report, 1997.
- B. R. Cullis and A. C. Gleeson. Spatial analysis of field experiments - an extension to two dimensions. *Biometrics*, 47:1449–1460, 1991.
- B. R. Cullis, A. C. Gleeson, W. J. Lill, J. A. Fisher, and B. J. Read. A new procedure for the analysis of early generation variety trials. *Applied Statistics*, 38:361–375, 1989.
- A. P. Dempster, M. R. Selwyn, C. M. Patel, and A. J. Roth. Statistical and computational aspects of mixed model analysis. *Applied Statistics*, 33:203–214, 1984.
- P. J. Diggle, P. J. Ribeiro, and O. F. Christensen. An introduction to model-based geostatistics. In Jesper Moller, editor, *Spatial Statistics and Computational Methods*, pages 43–86. Springer-Verlag, 2003.

- N. R. Draper and H. Smith. *Applied Regression Analysis*. John Wiley and Sons, New York, 3 edition, 1998.
- A. R. Gilmour, R. D. Anderson, and A. L. Rae. The analysis of binomial data by a generalised linear mixed model. *Biometrika*, 72:593–599, 1985.
- A. R. Gilmour, B. R. Cullis, and A. P. Verbyla. Accounting for natural and extraneous variation in the analysis of field experiments. *Journal of Agricultural, Biological and Environmental Statistics*, 2:269–293, 1997.
- A. R. Gilmour, B. R. Cullis, S. J. Welham, B. J. Gogel, and R. Thompson. An efficient computing strategy for prediction in mixed linear models. *Computational Statistics and Data Analysis*, 44:571–586, 2004.
- A. R. Gilmour, B.J. Gogel, B. R. Cullis, S. J. Welham, and R. Thompson. *ASReml User Guide Release 1.0*. VSN International, 5 The Waterhouse, Waterhouse St, Hemel Hempstead, HP1 1ES, UK, 2002.
- A. R. Gilmour, R. Thompson, and B. R. Cullis. AI, an efficient algorithm for REML estimation in linear mixed models. *Biometrics*, 51:1440–1450, 1995.
- A. C. Gleeson and B. R. Cullis. Residual maximum likelihood (REML) estimation of a neighbour model for field experiments. *Biometrics*, 43:277–288, 1987.
- H. Goldstein and J. Rasbash. Improved approximations for multilevel models with binary response. *Journal of the Royal Statistical Society, Series A*, 159:505–513, 1996.
- H. Goldstein, J. Rasbash, I. Plewis, D. Draper, W. Browne, M. Yang, G Woodhouse, and M Healy. *A user's guide to MLwiN*. Institute of Education, London, 1998. URL <http://multilevel.ioe.ac.uk/>.
- P. J. Green and B. W. Silverman. *Nonparametric regression and generalized linear models*. London: Chapman and Hall, 1994.
- W. R. Harvey. Least-squares analysis of data with unequal subclass frequencies. Technical Report ARS 20-8, USDA, Agricultural Research Service, 1960. Reprinted with corrections as ARS H-4, 1975.
- W. R. Harvey. *Users' guide to LSML76*. Ohio State University, Columbus, 1977.
- D.A. Harville and R.W. Mee. A mixed model procedure for analysing ordered categorical data. *Biometrics*, 40:393–408, 1984.
- K. A. Haskard. *Anisotropic Matérn correlation and other issues in model-based geostatistics*. PhD thesis, BiometricsSA, University of Adelaide, 2006.
- K. A. Haskard, B. R. Cullis, and A. P. Verbyla. Anisotropic matérn correlation and spatial prediction using reml. *Journal of Agricultural and Biological Sciences*, unknown:\*\*\*-\*\*\*, 2005.
- E. E. Kammann and M. P. Wand. Geoaddivitive models. *Applied Statistics*, 52(1): 1–18, 2003.

- A. Keen. Procedure IRREML. In *GLW-DLO Procedure Library Manual*, pages Report LWA-94-16, Wageningen, The Netherlands, 1994. Agricultural Mathematics Group.
- M. G. Kendall and A. Stuart. *The Advanced Theory of Statistics*, volume 1. Charles Griffin, London, 1969.
- M. G. Kenward and J. H. Roger. The precision of fixed effects estimates from restricted maximum likelihood. *Biometrics*, 53:983-997, 1997.
- P. W. Lane and J. A. Nelder. Analysis of covariance and standardisation as instances of prediction. *Biometrics*, 38:613-621, 1982.
- P. McCullagh and J. A. Nelder. *Generalized Linear Models*. Chapman and Hall, London, 2 edition, 1994.
- C.E. McCulloch and S. R. Searle. *Generalized, Linear, and Mixed Models*. Wiley, 2001.
- T. H. E. Meuwissen and Z. Luo. Computing inbreeding coefficients in large populations. *Genetics Selection Evolution*, 24, 1992.
- R.B. Millar and T.J. Willis. 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, 1999.
- J. A. Nelder. A reformulation of linear models. *Journal of the Royal Statistical Society, Series A*, 140:48-76, 1977.
- J. A. Nelder. The statistics of linear models: back to basics. *Statistics and Computing*, 4:221-234, 1994.
- H. D. Patterson and R. Thompson. Recovery of interblock information when block sizes are unequal. *Biometrika*, 58:545-54, 1971.
- J. C. Pinheiro and D. M. Bates. *Mixed-Effects Models in S and S-PLUS*. Berlin: Springer-Verlag, 2000.
- G. K. Robinson. That BLUP is a good thing: The estimation of random effects. *Statistical Science*, 6:15-51, 1991.
- German Rodriguez and Noreen Goldman. Improved estimation procedures for multilevel models with binary response: A case study. *Journal of the Royal Statistical Society, Series A*, 164(2):339-355, 2001.
- R. Schall. Estimation in generalized linear models with random effects. *Biometrika*, 78(4):719-727, 1991.
- S. R. Searle. *Linear Models*. New York: John Wiley and Sons, Inc., 1971.
- S. C. Self and K. Y. Liang. Asymptotic properties of maximum likelihood estimators and likelihood ratio tests under non-standard conditions. *Journal of the American Statistical Society*, 82:605-610, 1987.

- Michael L. Stein. *Interpolation of Spatial Data: Some Theory for Kriging*. Springer-Verlag, New York, 1999.
- M. M. Stevens, K. M. Fox, G. N. Warren, B. R. Cullis, N. E. Coombes, and L. G. Lewin. An image analysis technique for assessing resistance in rice cultivars to root-feeding chironomid midge larvae (diptera: Chironomidae). *Field Crops Research*, 66:25–26, 1999.
- W. W. Stroup, P. S. Baenziger, and D. K. Mulitze. Removing spatial variation from wheat yield trials: a comparison of methods. *Crop Science*, 86:62–66, 1994.
- R. Thompson. Maximum likelihood estimation of variance components. *Math. Operationsforsch Statistics, Series, Statistics*, 11:545–561, 1980.
- R. Thompson, B.R. Cullis, A. Smith, and A.R. Gilmour. 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, 2003.
- A. P. Verbyla. A conditional derivation of residual maximum likelihood. *Australian Journal of Statistics*, 32:227–230, 1990.
- A. P. Verbyla, B. R. Cullis, M. G. Kenward, and S. J. Welham. The analysis of designed experiments and longitudinal data by using smoothing splines (with discussion). *Applied Statistics*, 48:269–311, 1999.
- D. Waddington, S. J. Welham, A. R. Gilmour, and R. Thompson. Comparisons of some glmm estimators for a simple binomial model. *Genstat Newsletter*, 30:13–24, 1994.
- R. Webster and M. A. Oliver. *Geostatistics for Environmental Scientists*. John Wiley and Sons, Chichester, 2001.
- S. J. Welham. Glmm fits a generalized linear mixed model. In R.W. Payne and P.W. Lane, editors, *GenStat Reference Manual 3: Procedure Library PL17*,, pages 260–265. VSN International, Hemel Hempstead, UK, 2005.
- S. J. Welham, B. R. Cullis, B. J. Gogel, A. R. Gilmour, and R. Thompson. Prediction in linear mixed models. *Australian and New Zealand Journal of Statistics*, 46:325–347, 2004.
- S. J. Welham and R. Thompson. Adjusted likelihood ratio tests for fixed effects. *Journal of the Royal Statistical Society, Series B*, 59:701–714, 1997.
- R. D. Wolfinger. Heterogeneous variance-covariance structures for repeated measures. *Journal of Agricultural, Biological, and Environmental Statistics*, 1:362–389, 1996.
- Russ Wolfinger and Michael O’Connell. Generalized linear mixed models: A pseudo-likelihood approach. *Journal of Statistical Computation and Simulation*, 48:233–243, 1993.
- F. Yates. Complex experiments. *Journal of the Royal Statistical Society, Series B*, 2:181–247, 1935.

# Index

- \*Topic **anova**
  - wald.asreml, 92
- \*Topic **asreml**
  - asreml.control, 73
  - wald.asreml, 92
- >, 2
- %, 2
- AI algorithm, 13
- AI matrix, 16
- AIC, 17
- ainverse, 61
- Akaike Information Criteria, 17
- aliasing, 135
  - examples, 136
- anova
  - wald(), 26, 38
- aom, 17
- asreml
  - class, 26
  - methods, 77
  - object, 26, 75
- asreml, 75, 77
- asreml functions
  - asreml.read.table(), 24
- asreml functions
  - asreml(), 70
  - asreml.About(), 77
  - asreml.Ainverse(), 77
  - asreml.asUnivariate(), 79
  - asreml.constraints(), 80
  - asreml.control(), 73
  - asreml.man(), 81
  - asreml.read.table(), 81
  - asreml.variogram(), 82
- asreml methods
  - coef.asreml(), 83
  - fitted.asreml(), 83
  - plot.asreml(), 84
  - plot.asrVariogram(), 85
  - predict.asreml(), 85
  - residuals.asreml(), 87
  - summary.asreml(), 88
  - svc.asreml(), 89
  - tr.asreml(), 89
  - update.asreml(), 90
  - variogram.asreml(), 91
  - wald.asreml(), 92
- asreml() argument
  - fixed, 25
  - na.method.Y, 33
  - rcov, 25
- asreml() argument
  - data, 26
  - random, 25
  - rcov, 27, 32
- asreml.control, **73**, 77
- ASRe.ml.exe, 1
- asreml.object, **76**
- at(), 32
- Average Information, 1
- balanced repeated measures, 104
- Bayesian Information Criteria, 17
- BIC, 17
- BLUE, 14
- BLUP, 14, 15, 26
- coef(), 26
- coefficients
  - coef.fixed, 27
  - coef.random, 27

- coef.sparse, 27
- combining variance models, 15
- conditional distribution, 12
- conditional factor, 32
- conditional Wald statistics, 19
- conventions, 2
- correlated random effects, 15
- correlation
  - model, 11
- correlation models, 43
- covariance model, 11
  - isotropic, 10
- csv files, 24
- dense equations, 135
- diagnostics, 17
- direct product, 8, 10, 39
- distribution
  - conditional, 12
  - marginal, 12
- documentation, 2
- error variance
  - heterogeneity, 10
- estimation, 12
- expected information matrix, 13
- F statistics, 19
- fa( $\cdot$ , $k$ ), 52
- Fisher-scoring algorithm, 13
- fitted(), 26
- fixed
  - argument, 25
  - model formula, 27
- fixed effects, 8
- fixed terms, 27
  - covariates, 30
  - primary, 27
  - sparse, 30
- formula
  - fixed model, 27
- G structure, 11, 30, 39
- GLM, 33
- GLMM, 34
- help, 27
- heterogeneity
  - error variance, 10
  - heterogeneous variance models, 44
  - homogeneous variance models, 44
- hypothesis testing
  - random effects, 16
- identifiable, 15
- IID, 9
- incremental Wald statistics, 19
- information matrix
  - expected, 13
  - observed, 13
- initial values, 30
  - G.param, R.param arguments, 30
  - start.values agrument, 30
- inverse coefficient matrix, 26
- isotropic covariance model, 10
- kriging, 48
- kronecker product, 10
- likelihood
  - log residual, 13
  - residual, 12
- linear mixed model, 8
- longitudinal data
  - balanced example, 129
- marginal distribution, 12
- Matérn variance structure, 50
- matrix
  - average information, 26
  - inverse coefficient, 26
- MET, 10
- meta analysis, 10
- methods, 26
- missing values
  - covariates, 30, 33
  - design factors, 33
  - response, 33
- mixed
  - effects, 8
  - model, 8
- mixed model equations, 14
- model
  - correlation, 11
  - covariance, 11
- model formula
  - random, 30

- multi-environment trial, 10
- multivariate analysis, 35
- NA, 24
- na.method.Y, 33
- Nebraska Intrastate Nursery, 3
- nonidentifiable, 15
- objective function, 14
- observed information matrix, 13
- operator
  - +, 25
  - $\sim$ , 25
- ordering of terms, 135
- overspecified, 15
- P, 30
- parameter
  - scale, 8
  - variance, 8
- plot(), 26
- positive definite matrix, 45
- power, 139
- Predict
  - prwts, 67
- prediction, 63
- prior mean, 15
- product
  - direct, 10
  - kronecker, 10
- R structure, 10, 30, 39
- random
  - argument, 25
  - effects, 8
    - correlated, 15
  - model formula, 30
  - regression, 54
  - terms, 30
- random coefficients, 129
- random regression, 12, 54
- RCB, 30
  - design, 3
- rcov, 27, 32
  - argument, 25
- read.table(), 24
- REML, 12, 16
  - log-likelihood, 26
- REMLRT, 16
- resid(), 26
- residual
  - error, 8
  - likelihood, 12
- residual variance, 8
- residual maximum likelihood, 12
- scale parameter, 8
- score, 13
- section, 10
- separability, 10
- shrinkage, 15
- singularities, 135
- sparse equations, 135
- sparse matrix methods, 14
- spatial
  - analysis, 10, 110
  - example, 110
  - model, 42
- split plot, 95
- stratum variances, 22
- summary(), 26
- svc.asreml, 93
- tests of hypotheses, 16, 18
- trait, 35
- U, 30
- unbalanced data
  - sources of variability, 102
- unbalanced nested design, 99
- unreplicated trial, 115
- variance
  - parameter, 8, 12
  - estimation, 14
  - residual, 8
- variance model
  - specification, 39
- variance function
  - aexp(), 140
  - agau(), 140
  - ante( $k$ ), 140
  - ante( $k$ ), 52
  - ar1(), 32, 138
  - ar2(), 138
  - ar3(), 138
  - arma(), 139
  - chol( $k$ ), 140

- cir(), 140
- cor(), 139
- corb(), 139
- corg(), 139
- diag(), 140
- exp(), 139
- gau(), 139
- giv(), 141
- id(), 138
- idh(), 140
- ieuc(), 139
- iexp(), 139
- igau(), 139
- isp(), 140
- ma1(), 138
- ma2(), 138
- mtrn(), 140
- ped(), 141
- sar(), 138
- sar2(), 138
- str(), 54, 132, 141
- us(), 140
- variance model
  - combining, 15, 54
  - correlation, 43
  - heterogeneous, 44
  - homogenous, 44
- variogram, 18
- Wald statistics, 38
- Wald test, 19
- wald(), 26
- wald.asreml**, **92**
- weighted analysis, 32
- weights, 32