ASReml-R Reference Manual Version 4.2

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

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

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

[2]VSN International Ltd., United Kingdom (formerly NSW Department of Primary Industries, Orange, Australia)

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

May 27, 2023

ASReml-R Reference Manual Version 4.2

D. G. Butler, B. R. Cullis, A. R. Gilmour, B. J. Gogel and R. Thompson

Published by

VSN International Ltd. 2 Amberside House, Wood Lane, Hemel Hempstead, HP2 4TP, UK.

email: asreml@vsni.co.uk

website: https://vsni.co.uk/

Copyright notice

Bibliographical reference

The correct bibliographical reference for this document is:

Butler, D.G., Cullis, B.R., Gilmour, A.R., Gogel, B.G. and Thompson, R. 2023. ASReml-R Reference Manual Version 4.2. VSN International Ltd., Hemel Hempstead, HP2 4TP, UK.

Author's email addresses

David G. Butler: david_butler@uow.edu.au

Brian R. Cullis: brian_cullis@uow.edu.au

Arthur R. Gilmour: arthur.gilmour@vsni.co.uk

Beverley G. Gogel: beverley_gogel@uow.edu.au

Robin Thompson: robin.thompson@rothamsted.ac.uk

Companion documents

An additional document, the ASReml-R package reference, complements this reference manual. The ASReml-R package reference is a pdf version of the ASReml-R help pages obtained in R by typing `help(asreml)`. This, and other resources, are also available at the VSN International website https://asreml.kb.vsni.co.uk/article-categories/asreml-r-resources/.

Acknowledgements

The authors gratefully acknowledge the Grains Research and Development Corporation of Australia for their financial support. We thank the University of Wollongong (Australia) and Queensland Department of Agriculture and Fisheries (Australia) for permitting this research to be undertaken and for providing a stimulating environment for applied biometrical consulting and research. We sincerely thank Sue Welham for her contribution toward research into the use of linear mixed models and the development of appropriate software. Ian White is acknowledged for suggesting a convenient way of computing approximate variances of functions of variance parameters.

We also acknowledge the contributions of many members from the VSN International team, including Darren Murray, Martyn Byng, Giovanni Galli and Salvador Gezan, and many of the innumerable users that have contributed with suggestions, detected bugs and provided additional ideas to improve the library.

Contents

Preface

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

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 features of ASReml-R include:

- A flexible syntax for specifying variance models for the random effects, and the scope this offers the user. Users should be aware of the dangers of either overfitting or attempting to fit inappropriate variance models to small or highly unbalanced data sets. We stress the importance of the use of data driven diagnostics and encourage the user to read the examples chapter, in which we have attempted to not only present the syntax of ASReml-R in the context of real analyses but also to indicate some of the modelling approaches we have found useful.
- The REML routines use the Average Information (AI) algorithm, and sparse matrix methods for fitting the mixed model. This enables ASReml-R to efficiently analyse large and complex data sets.

This manual consists of seven chapters. Chapter 1 introduces ASReml-R, describes the conventions used throughout the manual, and describes the various data sets used for illustration; Chapter 2

presents a 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 5 describes special functions and methods for genetic analyses; Chapter 6 outlines the prediction of linear functions of fixed and random effects in the linear mixed model and Chapter 7 presents a comprehensive and diverse set of worked examples. In addition, the Appendix provides additional details on some of the options and functions that can help with specific analyses.

There are a number of new developments in Version 4.2. These include improvements in speed and memory management, together with additional options in some of the library functions.

- Memory access has been increased to a maximum of 96 GB, enabling you to analyse larger problems in less time.
- ASReml-R 4.2 has been optimised for multi-threaded processing, enabling you to use up to 16 threads.
- Many of ASReml-R 4.2 core routines have been reworked making it much faster than the previous version.
- Stability and optimisation have been enhanced to facilitate consistent results for future releases.

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. Additional data sets, particularly for larger examples, can be easily accessed from the R library `ASRdata` which can be downloaded from https://github.com/asrtools/asrdata.

We have extensively tested the software but it is inevitable that bugs will exist. These may be reported to support@vsni.co.uk. The authors and developers would also appreciate being informed of errors and requests of improvements to the manual and software.

Chapter 1

Introduction

1.1 What ASReml-R can do

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

- (Un)balanced longitudinal data.
- Repeated measures data (including multivariate analysis and trajectory-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-SA (Gilmour et al. 2002). The computational efficiency of ASReml-R arises from using the Average Information REML algorithm (giving quadratic convergence) and sparse matrix operations. However, because of overheads inherent in R language implementations, some very large problems may need to use the ASReml-SA program to overcome memory limitations.

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

1.2 Getting started

1.2.1 Installation

Installation instructions are provided on the VSN International website https://asreml.kb.vsni.co.uk/knowledge-base/asreml-r-installation-guide/

1.2.2 Help and references

Documentation for the `asreml()` function, support functions and related methods are available via the standard help mechanism; that is, `help(asreml)` or `?asreml` will display the ASReml-R documentation in text or HTML form depending on implementation and help system state.

A complete description of the components of an `asreml` object and the data sets used in this manual are given in the ASReml-R Package Reference, which is a pdf version of the ASReml help pages obtained in R by typing `help(asreml)`. This document is also available at the VSN International website. For additional resources, we encourage you to visit https://asreml.kb.vsni.co.uk/article-categories/asreml-r-resources/.

1.2.3 Using this manual

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

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

Prediction from the fitted linear mixed model is discussed in Chapter 6. In Chapter 7 you will find a wide range of additional worked examples. In the Appendixes, tables with descriptions and details of some options of functions and modules are presented with further relevant details.

Once ASReml-R is loaded, the data sets can be directly accessed via their names due to the `LazyData` characteristics of the library. For example, the `grass` data can be accessed using:

```
grass
```

or via the conventional call with the `data()` function of R:

```
data(grass)
```

The first few lines of the `grass` data are:

```
head(grass)
```

```
  Tmt Plant   y1   y3   y5    y7   y10
1 MAV     1 21.0 39.7 47.0  53.0  55.0
2 MAV     2 32.0 59.5 63.5  65.0  67.6
3 MAV     3 35.5 54.6 58.0  61.5  61.5
4 MAV     4 33.5 41.0 48.0  57.0  58.0
5 MAV     5 31.5 45.3 62.0 104.0 104.0
6 MAV     6 27.0 43.3 56.4  74.5  62.0
```

1.3 Data sets used

Below we list the data sets used in this manual.

Table 1.1: Data sets available in ASReml-R library

Name	Description
`binnor`	Footrot scores on lambs
`coop`	Coopworth lamb data
`ebay`	Ebay bids
`grass`	Plant height disease study (multivariate form)
`grassUV`	Plant height disease study (univariate form)
`harvey`	Weights of Hereford cattle
`harvey.ped`	Pedigree of Hereford cattle
`harveyg.ped`	Pedigree of Hereford cattle (with genetic groups)
`lamb`	Footrot counts on lambs
`met`	Raw wheat data from 8 locations with spatial information
`naf`	Barley study (naf)
`naf31H`	Marker scores barley study (naf)
`naf32H`	Marker scores barley study (naf)
`nassau`	Loblolly pine clone trial
`nassau.grm`	Loblolly genetic relationship matrix (GRM)
`nassau.snp`	Loblolly genetic marker scores
`nin89`	Wheat advanced breeding trial
`oats`	Oats variety-nitrogen factorial study
`orange`	Orange tree circumference
`owt`	Orange wether wool trial
`pixel`	Repeated CT scans on dogs
`rats`	Reproductive study on rats
`rice`	Rice resistance to bloodworms (univariate form)
`riceMV`	Rice resistance to bloodworms (multivariate form)
`shf`	Slate Hall farm wheat variety trial
`voltage`	Voltage regulators
`wheat`	Wheat variety trial
`wolfinger`	Growth curve study on rats

Chapter 2

Some theory

2.1 The general linear mixed model

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

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

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

2.1.1 Sigma parameterization of the linear mixed model

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

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

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

$$var(\boldsymbol{y}) \quad = \quad \boldsymbol{Z}\boldsymbol{G}(\boldsymbol{\sigma}_g)\boldsymbol{Z}' + \boldsymbol{R}_v(\boldsymbol{\sigma}_r) \tag{2.3}$$

which we will refer to as the *sigma parameterization* of the $\boldsymbol{G}$ and $\boldsymbol{R}$ variance structures, and the individual variance structure parameters in $\boldsymbol{\sigma}_g$ and $\boldsymbol{\sigma}_r$ will be referred to as *sigmas*. The variance models given by $\boldsymbol{G}$ and $\boldsymbol{R}_v$ are referred to as G *structures* and R *structures*, respectively.

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

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

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

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

2.1.2 Partitioning the fixed and random model terms

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

2.1.3 G structure for the random model terms

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

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

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

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

Example 2.2. Variance components mixed models. Building example 2.1 to a linear mixed model with more than one ($b > 1$) random effect (typically known as a variance components mixed model), the random effects $\boldsymbol{u}_i$ in $\boldsymbol{u}$, and the residual errors $\boldsymbol{e}$, are assumed pairwise uncorrelated and to each be normally distributed with mean zero and variance given by

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

and

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

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

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

2.1.4 Partitioning the residual error term

As for the fixed and random model terms, it is often useful or appropriate to consider a partitioning of the vector of residual errors $\boldsymbol{e}$ according to some conditioning factor. We use the term *section* to describe this partitioning and the most common example of the use of sections in $\boldsymbol{e}$ is when we wish to allow sections in the data to have different variance structures. For example, in the analysis of multi-environment trials (METs) it is natural to expect that each trial will require a separate (possibly spatial) error structure. In this case, for s sections we have $\boldsymbol{e} = [\boldsymbol{e}_1', \boldsymbol{e}_2', \ldots, \boldsymbol{e}_s']'$ assuming that the data vector is ordered by section, and where $\boldsymbol{e}_j$ represents the vector of errors for the j^{th} section.

2.1.5 R structure for the residual error term

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

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

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

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

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

2.1.6 Gamma parameterization for the linear mixed model

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

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

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

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

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

In this case

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

which we will refer to as the *gamma parameterization*, and the individual variance structure parameters in $\boldsymbol{\gamma}_g$ and $\boldsymbol{\gamma}_r$ will be referred to as *gammas*. ASReml-R switches between the sigma and gamma parameterizations for estimation. This is discussed further in Section 4.7.

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

2.1.7 Parameter types

Each sigma in $\boldsymbol{\sigma}_g$ and $\boldsymbol{\sigma}_r$ and each gamma in $\boldsymbol{\gamma}_g$ and $\boldsymbol{\gamma}_r$ has a parameter type, for example, variance components, variance component ratios, autocorrelation parameters, and/or factor loadings. Furthermore, the parameters in $\boldsymbol{\sigma}_g$, $\boldsymbol{\sigma}_r$, $\boldsymbol{\gamma}_g$ and $\boldsymbol{\gamma}_r$ can span multiple types. For example, the spatial

analysis of a simple column trial would involve variance components (sigma parameterization) or variance component ratios (gamma parameterization) and spatial autocorrelation parameters.

2.1.8 Variance structures for the random model terms

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

2.1.9 Variance models for terms with several factors

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

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

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

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

(see next section for the mathematical definition of a direct product structure $\otimes$), and

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

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

- $\boldsymbol{\Sigma}_{u_{AB}} = \boldsymbol{I}_A \otimes \boldsymbol{\Sigma}_B$ where $\boldsymbol{\Sigma}_B$ has an unstructured form. This means that the elements of $\boldsymbol{u}_{iB}$ are IID $N(\mathbf{0}, \boldsymbol{\Sigma}_B)$. This model is widely used in the analysis of multivariate data.
- $\boldsymbol{\Sigma}_{u_{AB}} = \boldsymbol{\Sigma}_A \otimes \boldsymbol{I}_B$ where $\boldsymbol{\Sigma}_A$ represents a first order autoregressive process. This means that the elements of $\boldsymbol{u}_{Ak}$ are IID realisations of this process. This model is used widely in the analysis of field trial data to model the spatial trend in one direction.

Example 2.3. A simple separable structure. If factor `A` has 3 levels and `B` has 2 levels, then the vector $\boldsymbol{u}_{AB}$ for the term `A:B` would be

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

Let $\boldsymbol{\Sigma}_{u_{AB}} = \boldsymbol{\Sigma}_A \otimes \boldsymbol{\Sigma}_B$ where

$$\boldsymbol{\Sigma}_A = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \text{ and } \boldsymbol{\Sigma}_B = \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix}.$$

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

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

2.1.10 Direct product structures

For the matrix $\boldsymbol{A} = \begin{bmatrix} a_{11} & \dots & a_{1p} \\ \vdots & \ddots & \vdots \\ a_{m1} & \dots & a_{mp} \end{bmatrix}$ and a second matrix $\boldsymbol{B}$,

the *direct product structure* of $\boldsymbol{A}$ and $\boldsymbol{B}$ is written in full as

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

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

2.1.11 Direct products in R structures

Separable structures occur naturally in many practical situations. Consider a vector of common errors associated with an experiment. The usual least squares assumption (and the default in ASReml-R) is that these are independently and identically distributed (IID). However, if $\boldsymbol{e}$ was from a field experiment laid out in a rectangular array of r rows by c columns, we could arrange the residuals as a matrix and might consider that they were autocorrelated within rows and columns. Writing the residuals as a vector in field order, that is, by sorting the residuals by rows within columns (plots within blocks) the variance of the residuals might then be

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

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

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

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

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

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

2.1.12 Direct products in G structures

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

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

where $\boldsymbol{\Sigma}^{(s \times s)}$ is the variance matrix for sites. This would imply that the varieties are independent random effects within each site, have different variances at each site, and are correlated across sites.

Important Whenever a random term is formed as the interaction of two factors you should consider whether the IID assumption is sufficient or if a direct product structure might be more appropriate. See Chapter 4 for details on specifying separable G structures in ASReml-R.

2.1.13 Range of variance models for R and G structures

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

Among the range of correlation models are:

- identity (that is, independent and identically distributed with variance 1)
- autoregressive (order 1 or 2)
- moving-average (order 1 or 2)
- autoregressive-moving-average, ARMA(1, 1)
- uniform
- banded
- general correlation.

Among the range of covariance models are:

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

There is also the facility to define models based on genetic relationship matrices, including additive relationship matrices generated by pedigrees or considering user-specified variance (or correlation) matrices. Further details are presented in Chapter 5.

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

2.2 Estimation

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

2.2.1 Estimation of the variance parameters

Estimation of the variance parameters is carried out using residual or restricted maximum likelihood (REML), developed by Patterson and Thompson (1971). An historical development of the theory can be found in Searle et al. (1992). Note firstly that

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

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

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

$$\boldsymbol{L}_1'\boldsymbol{X} = I_p, \quad \boldsymbol{L}_2'\boldsymbol{X} = \boldsymbol{0}$$

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

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

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

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

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

Estimation of the vector of variance parameters $\boldsymbol{\kappa} = [\boldsymbol{\sigma}_g'\ \boldsymbol{\sigma}_r']'$ is based on the log-residual likelihood,

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

where

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

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

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

$$\ell_R \;=\; -\frac{1}{2}(\log|\boldsymbol{C}| + \log|\boldsymbol{R}_v| + \log|\boldsymbol{G}| + \boldsymbol{y}'\boldsymbol{P}\boldsymbol{y}) \tag{2.11}$$

We can also write

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

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

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

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

The elements of the observed information matrix are

$$\begin{aligned} -\frac{\partial^2\ell_R}{\partial\kappa_i\partial\kappa_j} \;&=\; \frac{1}{2}tr(\boldsymbol{P}\boldsymbol{H}_{ij}) - \frac{1}{2}tr(\boldsymbol{P}\boldsymbol{H}_i\boldsymbol{P}\boldsymbol{H}_j) \\ &\qquad + \boldsymbol{y}'\boldsymbol{P}\boldsymbol{H}_i\boldsymbol{P}\boldsymbol{H}_j\boldsymbol{P}\boldsymbol{y} - \frac{1}{2}\boldsymbol{y}'\boldsymbol{P}\boldsymbol{H}_{ij}\boldsymbol{P}\boldsymbol{y} \end{aligned} \tag{2.13}$$

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

The elements of the expected information matrix $\boldsymbol{I}(\boldsymbol{\kappa},\boldsymbol{\kappa})$ are

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

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

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

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

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

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

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

The $\mathcal{I}_A$ matrix is the (scaled) residual sums of squares and products matrix of $\boldsymbol{y} = [\boldsymbol{y}_1, \ldots, \boldsymbol{y}_k]$, where $\boldsymbol{y}_i$ is the *working* variate for $\boldsymbol{\kappa}_i$ and is given by

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

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

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

2.2.2 Estimation/prediction of the fixed and random effects

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

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

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

Differentiating with respect to $\boldsymbol{\tau}$ and $\boldsymbol{u}$ leads to the mixed model equations (Robinson 1991) which are given by

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

These can be written as

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

where $\boldsymbol{C} = \boldsymbol{W}'\boldsymbol{R}_v^{-1}\boldsymbol{W} + \boldsymbol{G}^*$, $\tilde{\boldsymbol{\beta}} = [\hat{\boldsymbol{\tau}}' \;\; \tilde{\boldsymbol{u}}']'$ and

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

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

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

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

2.3 What are BLUPs?

Consider a balanced one-way classification. For data records ordered r repeats within b treatments regarded as random effects, the linear mixed model is $\boldsymbol{y} = \boldsymbol{X}\boldsymbol{\tau} + \boldsymbol{Z}\boldsymbol{u} + \boldsymbol{e}$ where $\boldsymbol{X} = \mathbf{1}_b \otimes \mathbf{1}_r$ is the design matrix for $\boldsymbol{\tau}$ (the overall mean), $\boldsymbol{Z} = \boldsymbol{I}_b \otimes \mathbf{1}_r$ is the design matrix for the b (random) treatment effects u_i, and $\boldsymbol{e}$ is the error vector with $\boldsymbol{e} \sim N(\mathbf{0}, \sigma^2 \boldsymbol{I}_{rb})$. Assuming that the treatment effects are random implies that $\boldsymbol{u} \sim N(\boldsymbol{A}\boldsymbol{\psi},\ \sigma_b^2 \boldsymbol{I}_b)$, for some design matrix $\boldsymbol{A}$ and parameter vector $\boldsymbol{\psi}$. It can be shown that

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

where $\bar{\boldsymbol{y}}$ is the vector of treatment means 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 $\boldsymbol{A}\boldsymbol{\psi}$. If $\boldsymbol{\psi} = \mathbf{0}$, the BLUP in (2.18) becomes

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

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

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

2.4 Inference: fixed effects

2.4.1 Introduction

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

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

$$\mathcal{W} = (\boldsymbol{L\hat{\tau}} - \boldsymbol{l})'\{\boldsymbol{L}(\boldsymbol{X}'\boldsymbol{H}^{-1}\boldsymbol{X})^{-1}\boldsymbol{L}'\}^{-1}(\boldsymbol{L\hat{\tau}} - \boldsymbol{l}) \tag{2.20}$$

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

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

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

2.4.2 Incremental and conditional Wald statistics

The basic tool for inference is the Wald statistic defined in equation (2.20). However, there are several ways $\boldsymbol{L}$ can be defined to construct a test for a particular model term, two of which are available in ASReml-R. An F-statistic is obtained by dividing the Wald statistic by r, the numerator degrees of freedom. In this form it is possible to perform an approximate F-test if we can deduce the denominator degrees of freedom. For balanced designs, these Wald F-statistics are numerically identical to the F-tests obtained from the standard analysis of variance.

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

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

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

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

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

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

and

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

Implicit in these calculations is that

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

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

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

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 the author 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

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

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

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

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

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

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

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

The preceding table includes a *marginality* or `M` code column 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

$$\mathsf{y} \sim 1 + \mathsf{Region} + \mathsf{Region:Site}$$

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

$$\mathsf{y} \sim 1 + \mathsf{Region} + \mathsf{Site}$$

with `Site` identified across `Region` rather than within `Region`. Then the nested structure is hidden but ASReml-R will still detect the structure and produce a valid conditional Wald F-statistic. This situation will be flagged in the `M` code field by changing the letter to lower case. Thus, in the nested model, the three `M` codes would be `.`, `A` and `B` because `Region:Site` is obviously an interaction dependent on `Region`. In the second model, `Region` and `Site` appear to be independent factors so the initial `M` codes are `.`, `A` and `A`. However they are not independent because `Region` removes additional degrees of freedom from `Site`, so the `M` codes are changed 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()`. The basic table includes the numerator degrees of freedom (denoted ν_{1i}) and the incremental Wald F-statistic for each term. To this is added the conditional Wald F-statistic and the `M` code if `ssType = "conditional"`.

2.4.3 Kenward and Roger adjustments

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

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

2.4.4 Approximate stratum variances

The stratumVariances method generated after running `wald()` returns approximate stratum variances and degrees of freedom for simple variance components models.

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

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

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

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

Thence

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

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

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

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

2.5 Inference: random effects

2.5.1 Tests of hypotheses: variance parameters

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

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

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

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

$$D = 2\log(\ell_{R2}/\ell_{R1}) = 2\left[\log(\ell_{R2}) - \log(\ell_{R1})\right] \tag{2.21}$$

which is strictly positive. If r_i is the number of parameters estimated in model i ($i = 1, 2$), 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 a hypothesis with the parameter on the boundary of the parameter space. It can be shown that for a single variance component, the theoretical asymptotic distribution of the REMLRT is a mixture of χ^2 variates, where the mixing probabilities are 0.5, one with 0 degrees of freedom (spike at 0) and the other with 1 degree of freedom. The approximate p-value for the REMLRT statistic (D), is 0.5(1-$\Pr(\chi^2_1 \leq d)$) where d is the observed value of D. This has a 5% critical value of 2.71 in contrast to

the 3.84 critical value for a χ^2 variate with 1 degree of freedom. The distribution of the REMLRT for the test that k variance components are zero, or tests involved in random regressions, which involve both variance and covariance components, involves a mixture of χ^2 variates from 0 to k degrees of freedom. See Self and Liang (1987) for details.

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

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

$$\begin{aligned} \text{AIC} &= -2\ell_{Ri} + 2t_i \\ \text{BIC} &= -2\ell_{Ri} + t_i \log \nu \end{aligned} \tag{2.22}$$

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-R for examining the adequacy of the assumed variance matrix for either R or G structures, or for examining the distributional assumptions regarding $\boldsymbol{e}$ or $\boldsymbol{u}$. Firstly, we note that the E-BLUP of the residual vector is given by

$$\begin{aligned} \tilde{\boldsymbol{e}} &= \boldsymbol{y} - \boldsymbol{W}\tilde{\boldsymbol{\beta}} \\ &= \boldsymbol{R}_v \boldsymbol{P} \boldsymbol{y} \end{aligned} \tag{2.23}$$

It follows that

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

The matrix $\boldsymbol{W}\boldsymbol{C}^{-1}\boldsymbol{W}'$ (under the sigma parameterization) is the so-called *extended hat* matrix. ASReml-R includes the σ^2 in this hat matrix under the gamma parameterization. It is the linear mixed effects model analogue of $\sigma^2\boldsymbol{X}(\boldsymbol{X}'\boldsymbol{X})^{-1}\boldsymbol{X}'$ for ordinary linear models. The diagonal elements are returned in the `hat` component of the `asreml` object.

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

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

Note `aom = TRUE` has not been validated for multivariate models or factor analytic models with zero specific variances.

2.5.3 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, $\boldsymbol{s}_i, i = 1, \ldots, n$, then the ordinates of the sample variogram are given by

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

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

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

ASReml-R also computes the variogram from predictors of random effects which appear to have a variance structure defined in terms of distance. The coordinates can be accessed using the `varioGram()` function, see the ASReml-R Package Reference https://asreml.kb.vsni.co.uk/article-categories/asreml-r-resources/for details.

Chapter 3

Fitting the mixed model

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

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

3.1 The data frame

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

Six rows of the NIN field trial data (available via `data(nin89)`) 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 242 data rows, one for each variety in each replicate.

```
    Variety Id  pid raw Rep nloc yield lat long Row Column
16   LANCER  1 1101 585   1    4 29.25 4.3 19.2  16      1
17    BRULE  2 1102 631   1    4 31.55 4.3 20.4  17      1
18  REDLAND  3 1103 701   1    4 35.05 4.3 21.6  18      1
19     CODY  4 1104 602   1    4 30.10 4.3 22.8  19      1
20 ARAPAHOE  5 1105 661   1    4 33.05 4.3 24.0  20      1
21  NE83404  6 1106 605   1    4 30.25 4.3 25.2  21      1
```

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 internally created from a text file source using an R function call like:

```
nin89 <- read.table(file = "/path/nin89.csv", header = TRUE, sep = ",")
```

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

- Blank lines are ignored.
- It is sensible to include a header line in the data file; if no header line is included, the columns are labelled $V1 \ldots V_n$ where n is the number of columns.
- The same column name should not be repeated. The numerals 1, 2, etc. are appended to subsequent repeated column labels.
- `NA` is the default 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 will cause an error.

Character fields such as `Variety` above are not always 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. This function reads data from a text file and automatically converts variates whose names begin with a capital letter in the header line into factors. So, for the NIN data we have

```
nin89 <- asreml.read.table(file = "/path/nin89.csv", header = TRUE, sep = ",")
```

which 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 = "/path/nin89.csv", header = TRUE, sep = ",")
nin89$Id <- as.factor(nin89$Id)
nin89$Rep <- as.factor(nin89$Rep)
nin89$Row <- as.factor(nin89$Row)
nin89$Column <- as.factor(nin89$Column)
```

3.2 Introducing the `asreml()` function call

A complete `asreml()` function call for a simple randomized complete block (RCB) design analysis of the NIN yield data are

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

where `nin89.asr` is the name we have chosen for the returned object, and `idv()` is the homogeneous identify variance structure (assumed by default). The key elements of this call are outlined below while the components of the returned object are described in Section 3.3.

3.2.1 Model formulae: specifying the linear mixed model

The linear model is specified in the `fixed` (required), `random` (optional) and `residual` (optional error component) arguments as formula objects. A third optional model argument `sparse` is also available but is not used explicitly (see also Section 3.11) in this example.

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

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

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

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

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

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

3.2.2 Finding the data

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

3.3 Components of the fitted model: the `asreml` object

A call to `asreml()` produces an object of class `asreml` which contains numerous components of the fit, some of these are

- The REML log-likelihood.
- Best linear unbiased predictors (BLUPs) of the random effects.
- Generalized least squares estimates of the fixed effects (or BLUEs).
- 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` output object is given in the ASReml-R Package Reference which is a pdf version of the ASReml-R help pages obtained in R by typing `help(asreml)`.

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. Also an `anova` type method is implemented by `wald()` (see Section 3.14).

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

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

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

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

3.4 A note on data order

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

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

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

3.5 Fixed terms

3.5.1 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 Appendix B.1. The fixed formula must contain at least one term which may simply be the intercept. By default the intercept is included in the fixed model; for example,

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

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

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

An intercept-only fixed model is specified by including only a `1` after $\sim$, for example,

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

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

```
asreml(
  fixed = y ~ lin(Row),
  ...
)
```

Model functions also exist to generate orthogonal polynomials (`pol()` and `leg()`) or splines (`spl()`) and to fit terms conditionally (`at()`; see Appendix B.1 and Section 3.7). Note that argument `fixed` is the only model formula where the response may be specified.

3.5.2 Sparse fixed terms

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

3.5.3 Covariates

For analysis purposes it is recommended that covariates be centred and rescaled to have a mean of 0 and a variance of 1 to avoid failure by potential 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$linrow2 <- scale(as.numeric(nin89$Row), center = TRUE, scale = FALSE)
```

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.

3.6 Random terms

The `random` model formula specifies the factors, interactions, covariates and special terms that comprise the random component of the model. All of these effects are estimated using sparse matrix methods. Each random term will have a variance model associated with it which, when no variance model function is specified, defaults to a scaled identity $\gamma \boldsymbol{I}_n$ or $\sigma^2 \boldsymbol{I}_n$ where γ is a variance ratio, depending on whether a sigma or gamma parameterization is used for fitting. This is equivalent to `idv()` and further details are presented in {sec:var}.

3.6.1 Initial values and constraints for variance parameters

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

For example, in the simple RCB field trial analysis

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

the gamma parameterization is used and 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()`.

3.6.2 Replacing elements in an internal object

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

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

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

```
(iv <- nin89.start$vparameters.table)
```

```
  Component Value Constraint
1   Rep!Rep   0.1          P
2   units!R   1.0          P
```

Elements of this table can be set by the usual R replacement methods. The new initial value for `Rep` (specified as 0.5) can be used in `asreml()` with the `G.param` argument. That is,

```
iv$Value[1] <- 0.5
nin89.asr <- asreml(
  fixed = yield ~ Variety,
  random = ~idv(Rep),
  residual = ~units,
  na.action = na.method(x = "include"),
  data = nin89,
  G.param = iv
)
```

3.6.3 Editing an external text file

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

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

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

3.6.4 Specifying variance structures

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

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

(assuming the data are correctly ordered as `Row` within `Column`) in the call, where `ar1v()` and `ar1()` are part of 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 C.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.

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

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

A similar function `dsum()` is available only for the residual formula and specifies a variance model for $\boldsymbol{e}$ as a direct sum of `l` variance matrices, one for each level of the conditioning factor. An example was presented earlier in Section 2.1.5.

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

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

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

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

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

- If a model structure specified defines a residual matrix then a variance factor associated with the appropriate sectioning level is added to the specified model to generate a variance matrix.
- It prunes the levels for a section so that *only* the levels of factors defining the residual variance structure for that section are used in forming that variance structure.

3.7.1 Specifying the model using `dsum`

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

```
residual = ~dsum(~ar1v(Column):ar1(Row) | Site)
```

In the above case, the error variance would be fixed at `1.0` as we are specifying `ar1v` for the first term, but for the case below, the error variance will not be fixed and this will be estimated.

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

For the MET data with three sites, we could specify a scaled AR1×AR1 variance structure for sites 1 and 3, but a scaled ID×AR1 structure for site 2, using the code:

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

or

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

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

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

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

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

An error would be returned for

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

Error: Residual model overparameterized - structure has 2 variance models

The above message is appearing because now we have two variance models, and the variance component associated with `idv()` is redundant with the variance component from `ar1v`.

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

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

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

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

3.7.2 Specifying variance structures using dimensions

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

```
residual = ~dsum(~ar1v(6):ar1(12) | Site, levels = 1) +
            dsum(~idv(8):ar1(10) | Site, levels = 2) +
            dsum(~ar1v(18):ar1(9) | Site, levels = 3)
```

3.7.3 Providing starting values for `dsum`

The majority of the variance models accept some starting values. Within `dsum`, the starting values have to be provided for *each* of the sections, and within the specified variance model. In the following example, we assign the starting values for the spatial correlations of 0.9 and 0.5 to two different groups of sites, and in all cases we specify a residual variance of 12.0. Note the use of `init` to assign a single value or a vector of values.

```
residual = ~dsum(~ar1v(Column, init = c(0.9, 12)):ar1(Row, init = 0.9) +
                   idv(Column, init = 12):ar1(Row, init = 0.5) | Site,
                   levels = list(c(1, 3), c(2)))
```

The order of the starting values needs to match what is expected for that given structure.

3.8 Two rules for defining the residual error term

The following two rules are critical to remember when running models with ASReml-R:

- Rule 1 The number of effects in the residual term *must* be equal to the number of data units included in the analysis.
- Rule 2 Where a separable variance structure is specified, each combination of levels of the single model terms specifying this structure must uniquely identify one unit of the data. For example, in the spatial analysis of a trial comprising 4 replicates of 24 varieties arranged as a rectangular array of dimension 4 rows by 24 columns (rows are replicates), a, AR1 × AR1 variance structure for the residuals can be specified by the model term `ar1(column):ar1(row)`, where `column` and `row` are the appropriate columns in the data file. However, the number of data units must be the product of the number of levels for `row` by the number of levels for `column`; 96 in this case. If this is not the case, or if more than one unit is associated with some row-column combination, ASReml-R will return an error message and it will not be possible to use `ar1(column):ar1(row)` for residual error. If there are fewer than 96 units, and each row-column combination present is associated with one unit, then the data would need to be augmented by completing (padding out) the full rectangular array to allow for an appropriate analysis.

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

3.9 Diagnostic plots

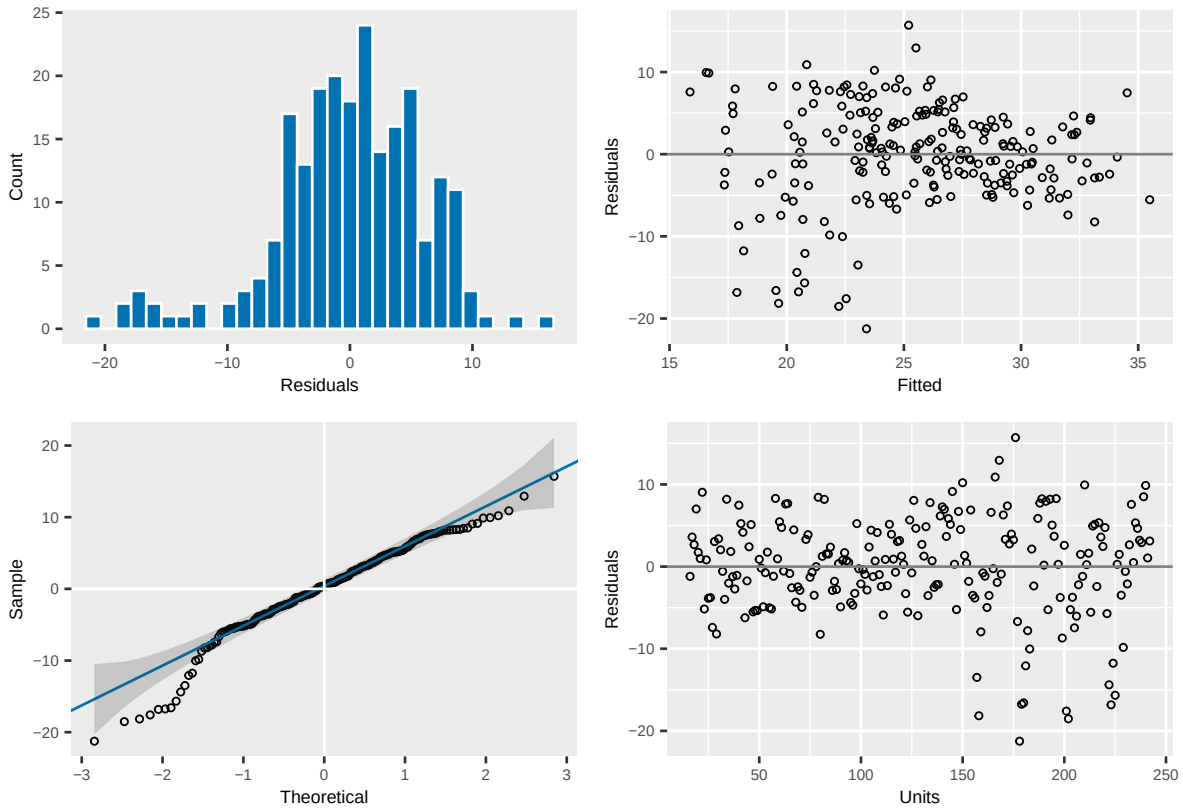
Several diagnostic statistics and plots can be used for assessing the quality of the fit, and to verify the validity of the assumptions made on the analysis fit. Of particular importance are the residual plots. The method `plot()` used with an `asreml` object will produce the following four plots:

- histogram of residuals: given that residuals are often assumed to be Normally distributed, this histogram should look approximately like a Normal distribution; this plots also helps to identify potential outliers as they will be on the tails of the histogram,

- Normal Q-Q plot: it can also be used to assess the assumption of Normality, where the residuals, under a perfect Normal distribution, will align exactly on the diagonal line; this plots helps to identify potential outliers that deviate considerably from this diagonal line,
- residuals against fitted values: this plots assists on verifying homogeneity of variances, if the variance is constant then residuals should lie within an uniform horizontal band; this plot helps to identify potential outliers as they will be outside this band, and
- residuals against unit number: this plot is useful to determine the approximate location of a given observation, but it can be useful to identify some structure on the data if observations are sorted by some relevant criteria.

To illustrate this, we will be using the NIN yield data arranged as an RCB. The asreml code is

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



An a composite plot of the diagnostic output described above can obtained with `plot(nin89.asr)`. Here, can see a few observations to the left that appear to be outside of the distribution. Further exploration is required to determine the reason. To explore the individual residual values it is possible to use `residuals()` with an `asreml` object.

The method `plot()` presents, by default, the raw or `working` residuals, but there are other options within `asreml`, further details can be found in the R help. If the residual structure of the model contains multiple sections, the default plots are conditioned on the factor whose levels define the sections. The library `asreml` has additional diagnostics that help to assess the distributional assumptions for the residuals and other random terms (see Section 2.5.2).

3.10 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 3.6.1, that is, editing a default R parameter list object with `asreml.gammas.ed()` (`start.values = TRUE`) or create and edit an external text file with `start.values = "filename"`, and changing the constraint of the units variance to `F`.
2. Set the units variance with the family argument

```
asreml(
  ...,
  weights = wt,
  family = asr_gaussian(dispersion = 1.0),
  ...
)
```

3.11 Missing values

Missing values in the response variable, and covariates or factors are allowed in ASReml-R, and these are treated as described below.

Missing values in the response

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

Missing values in the explanatory variables

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

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

3.12 Generalized linear (mixed) models

3.12.1 GLMs

The main function `asreml()` includes family functions for fitting Generalized Linear Models (GLM) (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 (using `dispersion = NA`). Table 3.1 lists the link functions that can be used to connect the linear predictor $\boldsymbol{\eta}$ to the mean ($\boldsymbol{\mu}$) on the data scale.

Table 3.1: 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 μ is the mean on the data scale, $\boldsymbol{\eta} = \boldsymbol{X\tau}$ is the linear predictor on the underlying scale and D is the default.

3.12.2 GLMMs

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 argument `family` is used

to define the distribution considered for the GLM or GLMM models. For example, the data frame `binnor` has a binary response named `score4` that corresponds to the incidence of footshape class 1. The corresponding GLMM model for this data are:

```
binnor.asr <- asreml(
  fixed = score4 ~ Sex + Grp,
  random = ~Sire,
  family = asr_binomial(link = "logit", dispersion = 1, total = NULL),
  data = binnor
)
```

The library `asreml()` has implemented the techniques penalized quasi-likelihood or PQL (Breslow and Clayton 1993). This technique can give somehow misleading results in certain situations, therefore we recommend to use GLMMs for advanced users and with care.

3.13 Multivariate analyses

Multivariate analyses are 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.

3.13.1 Model specification

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

```
asreml(
  fixed = cbind(y1,..., yk) ~ trait,
  ...
)
```

In this case, `asreml()` creates an internal factor `trait` (the multivariate equivalent to the univariate general mean) with the names of the response variables 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 Orange wether trial data, the term `trait` is a factor generated by `asreml()` with $t = 2$ levels `gfw` and `fdiam`. Internally, `asreml()` expands the data frame by repeating each row t times such that traits are nested within experimental units.

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

The following code examples illustrate the specification of multivariate models in `asreml()`.

3.13.2 A bivariate example

The `asreml()` function call for a basic bivariate analysis of the Orange wether trial data are:

```
wether.asr <- asreml(
  fixed = 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,
  residual = ~id(units):us(trait, init = c(0.2, 0.2, 0.4)),
  data = owt
)
```

Final estimates of the variance components are given by `summary(wether.asr)$varcomp`.

3.13.3 A repeated measures example

Wolfinger (1996) summarizes a range of variance structures that can be fitted to repeated measures data, demonstrating the models using the rat data set. 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 + trait:Treatment,
  residual = ~id(units):us(trait, init = rep(0, 15)),
  maxit = 20,
  data = wolfinger
)
```

Note that 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 internally from the phenotypic variance-covariance matrix. The fitted variance components are given by `summary(wolfinger.asr)$varcomp`.

3.13.4 Specifying multivariate variance structures

A more sophisticated default error structure is required for multivariate analysis in ASReml-R. Using the notation of Chapter 4. Consider a multivariate analysis with t traits and n units in which the

data are ordered traits within units. An algebraic expression for the variance matrix in this case is

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

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

For a standard multivariate analysis we have the following important elements:

- The error structure must be specified as two-dimensional, with independent units and often an unstructured variance matrix across traits.
  - The `residual` for this model is therefore `residual = ~id(units):us(trait)`.
  - Missing values are allowed and *must* be fitted; `asreml()` automatically includes the special factor `mv` in the `sparse` formula in such cases.
- For the default analysis where the response is specified as a matrix, the R structure *must* reflect the data order of *traits within units* which means that the term `units` must appear before `trait` in the `residual` formula.
- Variance parameters are variances, not variance ratios.
- The error structure is often specified as an unstructured variance matrix but correlation models may also be used. `asreml()` attempts to detect such cases and fix or estimate the residual scale parameter accordingly.

  For example, with the Wolfinger data the time points 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 + trait:Treatment,
  residual = ~id(units):ar1v(trait),
  data = wolfinger
)
```

- 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 of the phenotypic variance-covariance matrix in forming initial values.

3.14 Testing of terms: the `wald()` method

The type of object returned by the `wald()` method depends on the option specified for 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 effect terms is returned. These are tested sequentially, which means that factors are adjusted for terms higher in the table (or not in the table), but ignoring terms that occur below.

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

Conditional F-statistics

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

The data frame has three 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 2.4 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-R will only attempt the calculation if specifically requested as it requires further iterations of the model (using internally `update()`).

As an example, for the Orange wether trial data we obtain the pseudo analysis of variance calculating the approximate denominator degrees of freedom and using the conditional F-tests by

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

Chapter 4

Specifying variance structures

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

- The mixed linear model
$$\boldsymbol{y} = \boldsymbol{X}\boldsymbol{\tau} + \boldsymbol{Z}\boldsymbol{u} + \boldsymbol{e}$$
has a residual term
$$\boldsymbol{e} \sim N(\mathbf{0},\ \boldsymbol{R}_v(\boldsymbol{\sigma}_r))$$
and random effects
$$\boldsymbol{u} \sim N(\mathbf{0},\ \boldsymbol{G}(\boldsymbol{\sigma}_g))$$
where, in the most complex forms
$$\boldsymbol{R}_v = \oplus_i \boldsymbol{R}_{v_i}$$
$$\boldsymbol{G} = \oplus_j \boldsymbol{G}_j$$
and each
$$\boldsymbol{R}_{v_i} = \boldsymbol{R}_{v_i}(\boldsymbol{\sigma}_{r_i})$$
$$\boldsymbol{G}_j = \boldsymbol{G}_j(\boldsymbol{\sigma}_{g_j})$$
where $\boldsymbol{\sigma}_{r_i}$ and $\boldsymbol{\sigma}_{g_j}$ parameterize the respective variance models for each section.
- We use the terms *R structure* and *G structure* to refer to the matrices $\boldsymbol{R}_{v_i}$ and $\boldsymbol{G}_j$ above in a syntactic manner, respectively.
- R and G structures are typically formed as a direct product of particular variance models.
- The order of terms in a direct product must agree with the order of factors 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 (homogeneous) variance form (*e.g.* `ar1v()`) and to a heterogeneous variance form (*e.g.* `ar1h()`).

- Variance components are estimated as gammas (relative to the overall scale parameter, σ_e^2) when the gamma parameterization is used.

Previously, in Chapter 2 some theoretical details were presented. Here, we begin this chapter by considering an ordered sequence of variance structures for the NIN variety trial (Section 4.1) as an introduction to variance modelling in practice. Following, we consider these topics in detail in Section 4.2.

Variance models are specified with special model functions in the `random` and `residual` formulae. Scaled identity defaults are used if no variance model is explicitly specified (*i.e.* `idv()`). Table C.1 presents the complete range of variance models available in ASReml-R and details of individual (variance model) function calls are given in Section 4.3. Most of the models listed in Table C.1 are correlation models 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.

Finally, rules for combining variance models and methods for setting initial values, together with constrains and functions of variance components are given in the last few sections.

4.1 A sequence of structures for the NIN field trial data

By way of introduction, six variance structures of increasing complexity are considered for the NIN field trial data. This is to give a general feel for variance modelling in ASReml-R from a practical perspective and some idea of the types of models that are possible (for a list see Table C.1). The sequence of variance structures considered for the NIN field trial data are summarized in Table 4.1).

This section illustrates:

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

For ease of exposition we first describe the models fitted and reported when the variance models are explicitly specified.

For the models in this section an alternative fitting algorithm based on a gamma parameterization has advantages. In Section 4.7 we give one of the RCB data model examples in this gamma parameterization, provide some alternative specifications and comment on the summaries. In Section 4.1.2 we also show how some specifications can be reduced.

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

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

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

Model 1b: RCB analysis with G and R structures

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

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

Note that when specifying G structures ASReml-R automatically adds a scale parameter if a correlation model is specified (see Section 4.1.2 for more details). Only one of the models specified in a G structure can be a variance model. If the variance matrix of a term contains several component matrices then the problem of *identifiability* arises.

For example, a random term

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

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

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

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

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

- The direct product structure is implied by the ":" operator. The order in which factors appear in the `residual` formula also specifies the order in which the data must be sorted. Because `Column` is specified before `Row`, the implication is that the data are in the order *rows within columns*. ASReml-R does not reorder the observations; if the data frame is not in the order specified by `residual` then an error is generated and it must be reordered outside `asreml()`.
- Using a separable model for the R structure implies that the data can be regarded as a matrix or array whose data are indexed by the levels of the factors that represent the rows and columns of this array. In this field trial example these factors are `Row` and `Column`, respectively. For this structure to be applicable, the data in this case must be augmented with 18 additional missing values. `Variety` is arbitrarily coded as `LANCER` for all of the extra missing plots.
- ASReml-R automatically includes missing values in the sparse component with a factor named `mv` (see Section 3.11).

Model 2b: two-dimensional spatial model

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

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

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

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

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

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

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

This model is equivalent to **2c** but with the spatial model defined as a G structure rather than an R structure. As we discussed in **1b**, when the G structure term involves more than one model, all but one of the models must be a correlation model (Section 4.4). In this example `ar1v()` is the variance model.

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

4.1.1 An alternative fitting algorithm based on the gamma parameterization

In all of the data models above, the residual specifies a variance model with a single variance parameter and an alternative fitting algorithm, based on the gamma parameterization, can avoid specification of an initial residual variance and lead to speedier convergence. This can easily be actioned before the call to `asreml()` by setting:

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

Note that this call will set the `gammaPar` to `TRUE` for the duration of the session unless it is reset to `FALSE`. Section 4.1.2 discusses various more succinct default options. Section 4.7 provides a series of simple RCB data examples in this gamma parameterization with some alternative more implicit default specifications, and comments on the summaries.

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
1a	`fixed = yield ~ Rep + Variety`	-	-	-	units	idv()	-

1b	`fixed = yield ~ Variety,`
	`random = ~idv(Rep),`
	`residual = ~idv(units)`
		Replicate	idv()	-	units	idv()	-

2a	`fixed = yield ~ Variety,` `residual = ~idv(Column):ar1(Row)`	-	-	-	Column:Row	idv()	ar1()
2b	`fixed = yield ~ Variety,` `residual = ~ar1v(Column):ar1(Row)`	-	-	-	Column:Row	ar1v()	ar1()

2c	`fixed = yield ~ Variety,`
	`random = ~idv(units),`
	`residual = ~ar1v(Column):ar1(Row)`
		units	idv()	-	Column:Row	ar1v()	ar1()

3	`fixed = yield ~ Variety,`
	`random = ~ar1v(Column):ar1(Row)`
	`residual = ~idv(units)`
		Column:Row	ar1v()	ar1()	units	idv()	-

4.1.2 Reducing the specification using defaults

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

Note that in cases when the residual variance model is not specified or is not completely specified and implies a scaled identity matrix, the gamma parameterization is used. So

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

and

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

are equivalent to

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

and

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

4.2 Types of variance models

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

4.2.1 Correlation models

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

$$\mathbf{\Sigma} = [\rho_{ij}] \quad : \quad \left\{ \begin{array}{ll} \rho_{ii} = 1, & \forall i \\ \rho_{ij} = \rho_{ji}, & |\rho_{ij}| \leq 1,\ i \neq j \end{array} \right\}$$

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

4.2.2 Homogeneous variance models

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

That is,

$$\mathbf{\Sigma} = [\sigma_{ij}] \quad : \quad \left\{ \begin{array}{ll} \sigma_{ii} = \sigma^2, & \forall i \\ \sigma_{ij} = \sigma_{ji}, & i \neq j \end{array} \right\}$$

Note that if $\mathbf{\Sigma}$ is a correlation model, a homogeneous variance model (with one extra parameter) is formed as $(\sigma^2 \boldsymbol{I})\mathbf{\Sigma}$. For example, the homogeneous variance model corresponding to `id()` is `idv()` where $\mathbf{\Sigma} = \sigma^2 \boldsymbol{I}_\omega$ (or $\mathbf{\Sigma} = \gamma \boldsymbol{I}_\omega$).

4.2.3 Heterogeneous variance models

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

$$\mathbf{\Sigma} = [\sigma_{ij}] \quad : \quad \left\{ \begin{array}{ll} \sigma_{ii} = \sigma_i^2, & i = 1 \ldots \omega \\ \sigma_{ij} = \sigma_{ji}, & i \neq j \end{array} \right\}$$

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

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

4.2.4 Positive definite matrices

Formation of the mixed model equations (MME) requires the inversion of the variance matrix in the R and G structures. Therefore normally this requires these matrices to be non-singular. The two exceptions are 1) the fa model which has been specifically designed to fit singular matrices (Thompson et al. 2003) and, 2) when singular known relationship matrices are used when ASReml-R takes account of the implied constraints in the singular relationship matrices.

4.3 Variance model functions

ASReml-R has a wide range of variance models that can be used to specify the variance matrix of terms in the `random` and `residual` formulae. The following considers the various models in terms of functional groups and describes their syntax and application. In general, the correlation models described in the following sections have corresponding variance models whose names are simply derived by appending `"v"` or `"h"` to the correlation function name. In the former case this yields a homogeneous variance model while the latter gives the corresponding heterogeneous model.

4.3.1 Default identity

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

Required arguments

`obj` a factor in the data frame.

Optional arguments

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

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

Details

ASReml-R uses the `id()` correlation model or the `idv()` simple variance component model, depending on context (see the rules for combining variance models in Section 4.4) for terms in the `random` or `residual` formulae that have no variance model explicitly specified.

4.3.2 Time series type models

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

`ma2(obj, init = NA)`

`arma(obj, init = NA)`

Description

Includes autoregressive models of order 1, 2 and 3 (`ar1`, `ar2` and `ar3`), the symmetric autoregressive (`sar`), the constrained autoregressive order 3 (`sar2`), moving average models of order 1 and 2 (`ma1`, `ma2`) and the autoregressive-moving average model (`arma`).

Required arguments

`obj` a factor in the data frame.

Optional arguments

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Description

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

Required arguments

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

`y` a field (continuous variate) in the data frame containing the y coordinates.

Optional arguments

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

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

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

`phi` the range parameter. Default: $\phi = NA$.

`nu` the smoothness parameter. Default: $\nu = 0.5$.

`delta` governs geometric anisotropy. Default: $\delta = 1.0$.

`alpha` governs geometric anisotropy. Default: $\alpha = 0.0$.

`lambda` specifies the choice of metric. Default: $\lambda = 2$ for Euclidean distance.

For the Matérn function, if an argument is numeric, it is treated as a starting value for estimation and given the constraint code `P` (positive). This behaviour can be altered by concatenating the numeric value followed by the constraint code (`P`, `U` or `F`) into a character string (for example, $\nu =$ `0.5U`). If an argument is absent from the call, the corresponding parameter is held fixed at its default value. Further details and description of the Matérn class is provided in Appendix C.2.1.

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, or
- 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:

- `unique(x)` where `x` is the required argument to the model function identifying the field in the data frame containing the points,
- the `dist` argument to the model function, or
- the `pwr.points` list argument to `asreml()`.

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

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

4.3.4 General structure models

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

Description

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`) and its variants (`sfa`, `facv`, `rr`).

Required arguments

`obj` a factor in the data frame.

Optional arguments

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

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

[1] `chol`, `cholc` and `ante` models have $(k+1)(n-k/2)$ parameters but $n(n+1)/2$ initial values. These are given row-wise in the lower triangle of an unstructured matrix 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 the order of factor analytic models (`fa`).

Details

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

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

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

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

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

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

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

For example, if a factor `Site` has four levels then `cholc(Site, 1)` generates $\boldsymbol{\Sigma} = \boldsymbol{LDL}'$ where

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

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

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

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

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

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

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

Considering the above example for a factor `Site` with four levels, here `ante(Site, 1)` generates $\boldsymbol{\Sigma}^{-1} = \boldsymbol{UDU}'$ where

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

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

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

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

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

For example, if `Site` is a factor with four levels, the component matrices for `facv(Site,1)` are

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

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

Alternatively the variance-covariance matrix $\mathbf{\Sigma}^{\omega\times\omega}$ can be defined in `sfa(, k)` using a *correlation scale* with

$$\mathbf{\Sigma} = \boldsymbol{DCD}$$

where $\boldsymbol{D}^{\omega\times\omega}$ is a diagonal matrix such that $\boldsymbol{DD} = \mathrm{diag}\,(\mathbf{\Sigma})$ *i.e.* a diagonal matrix with the diagonal elements of $\mathbf{\Sigma}$, and $\boldsymbol{C}^{\omega\times\omega}$ is a correlation matrix of the form $\boldsymbol{F}\boldsymbol{F}'+\boldsymbol{E}$, where $\boldsymbol{F}$ is a matrix of loadings on the correlation scale and $\boldsymbol{E}$ is a diagonal matrix and is defined by difference. Comparison of $\mathbf{\Sigma}$ under the two parameterization show that $\boldsymbol{DF} = \mathbf{\Gamma}$ and $\boldsymbol{DED} = \mathbf{\Psi}$.

The parameters for the `sfa(, k)` model are specified in the order of the loadings for each factor ($\boldsymbol{F}$) followed by the variances (the diagonal elements of $\mathbf{\Sigma}$ or $\boldsymbol{DD}$). Note that the parameters for the `facv(, k)` model are also ordered in this way.

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

Limitations

The functions `fa(, k)` and `sfa(, k)`, unlike `facv(, k)`, do not allow singular $\mathbf{\Sigma}$ matrices to be estimated. In addition, constraints are required in $\mathbf{\Gamma}$ for $k > 1$ for identifiability. These are automatically set unless the user ensures identifiability by constraining one parameter in the second column, two in the third column, etc. With `rotate.fa = FALSE` (the default), ASReml-R fixes the

$j = 1, ..., i-1$ loadings for the i-th factor ($2 \leq i \leq k$) to zero and their corresponding boundary constraints to F. The total number of constraints is $k(k-1)/2$.

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

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

Updating loadings in factor analytic models

The algorithm for updating loadings in factor analytic models has been improved in the starting with ASReml-R Version 4. The motivation for change was that the original update procedure sometimes produced unreasonable updates, or otherwise came near to convergence and then drifted away. The present procedure modifies the average information matrix by increasing the diagonal elements pertaining to loadings by a percentage, p. The default is to start with $p = 10\%$ and reduce it by 1 or 2% each iteration down to 1%. If the starting values are poor, 10% may not be a sufficient initial retardation. If it appears the updates are unreasonable, ASReml-R will increase the value of p by 10% and then continue. The user can set the initial value of p with the option `ai.penalty = p`. After the penalty has reduced to 1%, it is further reduced to 0.2%. The qualifier can be used to set p to 0 if desired.

Another option, `ai.loadings`, allows further control of the AI updates of loadings in extended factor analytic, `fa(, k)`, models. After ASReml-R calculates updates for variance parameters, it checks whether the updates are reasonable and sometimes reduces them over and above any `step.size` shrinkage. The extra shrinkage has two levels. Loadings that change sign are restricted to doubling in magnitude, and if the average change in magnitude of loadings is greater than 10-fold, they are all shrunk. Further, when `ai.loadings = n` is specified (default `n = -1` specifies no action) and the user has not imposed identifiability constraints, then ASReml-R imposes them using `ai.rotate = TRUE` and it also prevents AI updates of some loadings during the first n iterations. For $k > 1$ factors, only the last factor is estimated (conditional on the earlier ones) in the first $k-1$ iterations. Then pairs, including the last, are estimated until iteration n.

4.3.5 Special case of factor analytic: `rr()`

A special case of `fa(, k)` is the reduced rank factor analytic model of `rr(, k)`. Here, the variance matrix $\mathbf{\Sigma}^{\omega \times \omega}$ is modelled as

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

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

For example, if `Site` is a factor with four levels, the component matrix for `rr(Site, 1)` is simply

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

Note that for computational reasons there can only be one `rr()` term in a compound model term, and it is recommended that the `rr()` term is the first term. In a three-way structure with an `rr()` term, if a relationship structure (specified using `vm()`) is used then it should be the third term and only an identity term can be used for the second term.

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

4.3.6 Known relationship structures

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

`ide(obj, source)`

Required arguments

`obj` a factor in the data frame.

`source` The known inverse or relationship matrix. This can be:

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

`singG` This argument is ignored if `source` has class `ginv` or its `INVERSE` attribute is `TRUE`. This happens when `source` is one of the following cases:

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

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

- PD source is positive definite (default).
- ND source is non-singular indefinite (positive and negative roots). In this case, ASReml-R ignores the indefinite condition and proceeds.
- PSD source is positive semi-definite. In this case, ASReml-R proceeds using Lagrangian multipliers to process the matrix. Two cases arise: whether the singularity arises because of an effect has zero variance or whether it arises as a linear dependence. An example of the first is when the GRM represents a dominance matrix, and the list of genotypes includes fully inbred individuals which by definition have no dominance. An example of the second is when the list of genotypes includes clones.
- NSD source is singular indefinite (positive, zero and negative roots). The indefinite condition is ignored and ASReml-R proceeds using Lagrange multipliers as for PSD matrices.

Details

If `source` inherits from class `Matrix`, ASReml-R will convert `source` internally to either sparse triplet form (class `dsparseMatrix`), or dense vector form (class `ddenseMatrix`) for processing. The names of the levels of `vm()` are given by the `rowNames` attribute associated with `source`. The number of levels of `vm(obj, source)` might be greater than the levels of `obj` in the data frame.

The `ide(obj, source)` structure creates a term with the levels associated with `source`, and modelled by the homogeneous form of the identity variance structure. If an `ide()` term succeeds its partner `vm()` term then `source` can be omitted from `ide()` and ASReml-R uses the source from the partner `vm()`.

Important If a model term is specified only in terms of `vm()` then a variance component is associated to it. But, if the model term if composed by two elements and one is a `vm()` term, then at least one of the other model terms must then specify a variance explicitly, as opposed to a correlation matrix (for example, `vm(genotype):idv(Site)` instead of `vm(genotype):Site`).

Linking a relationship matrix to regressor variables

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

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

For example, the code below fits such a model and estimates the marker effects given that the matrix $\boldsymbol{K}$ is in the R object `K` and the original $p \times m$ matrix of marker scores is in the R object `M`.

```
K <- asreml::nassau.grm
M <- as.matrix(asreml::nassau.snp)
attr(K, "INVERSE") <- FALSE

nassau.asr <- asreml(
    fixed = ht6 ~ CultureID/Rep,
    random = ~vm(clonefv, K) + ide(clonefv) + Rep:IncBlock,
    residual = ~idv(units),
    data = nassau
)

nassau.mef <- meff(
    object = nassau.asr,
    mef = list(K = "M"),
    effects = ~vm(clonefv, K)
)
```

The reason for quoting the name `"M"` is so that when R is passing the arguments through to the `meff` function it will not evaluate the object (which is typically large), as this will cause issues with memory and speed.

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

4.3.7 Variance structures spanning several model terms

`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 value 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 random model terms. Sometimes it is appropriate, for example in random regression models, to include a covariance parameter. The model terms in `form` are kept together and identified by the first term in the sequence. The variance structure defined in `vmodel` begins at the first term and covers the subsequent terms in the sequence. The overall size of the variance model is checked against the total number of levels of the terms in `form`, however, the sequence of effects matching the variance structure definition is not checked.

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

```
str(~Subject +  time:Subject, ~us(2):id(10))
```

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

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

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

Here, the set of subject intercepts has a common variance (σ_{II}), and the set of subject slopes has a (different) common variance (σ_{SS}). Intercepts and/or slopes from two different subjects are independent, but the intercept and slope from any given subject have covariance σ_{IS} (or correlation

$\sigma_{IS}/\sqrt{\sigma_{II}\sigma_{SS}}$). In this context, we use integers as input to emphasize that the arguments are specifying the size of the variance structure. For this example, `id(10)` can be replaced by `id(Animal)`.

This random regression example has been developed to describe the form of the `str()` function, but note that this model is equivalent to

```
us(pol(time)):id(Subject)
```

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

Below we present an example for the Reduced Animal Model with the need to form a composite design matrix as the sum of half a sire and half a dam matrix. We first create in our data frame a variate with all values equal to 0.5.

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

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

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

Note that internally the `Loc` effects are extended by 1 to include the one extra factor in `fa(, 1)`, and the levels associated with `P.Male` and `P.Female` are extended to include all levels within `Ainv`.

We note that the functionality of `and()` allows the term `and(fa(Loc, 1):vm(P.Female, ainv):half)` to be written in the alternative form `and(fa(Loc, 1):vm(P.Female, ainv), 0.5)`.

4.4 Rules for combining variance models

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

- When combining variance models in both R and G structures, the resulting direct product structure must match the ordered effects with the outer factor first. For example, the NIN data are ordered rows within columns. This is why in **Model 3** (page 55) the `ar1v()` variance model for `Column` is specified first in the interaction term.
- ASReml-R automatically includes and estimates an error variance parameter for each section of the R structure that is a correlation matrix.

- When the G structure involves more than one variance model, one must be either a homogeneous or a heterogeneous variance model and the rest should be correlation models; if more than one are non-correlation models then constraints should be used to avoid identifiability problems, that is, to prevent attempts to estimate confounded parameters.

4.5 Default initial values and constraints for variance parameters

The default initial values are 0.1 for both variance ratios and correlations, and $0.1 * v$ for variance components, where v is half the 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 3.6.1 and particularly for complex or unbalanced analyses it is recommended that starting values are provided to facilitate convergency.

4.5.1 Providing initial values for variance structures

In ASReml-R, the majority of the variance models accept some starting values. Providing good starting values facilitate the convergency of the mixed model, and for large and complex models, the model fit might finish quicker. Therefore, we recommend its use in most cases, particularly with unbalanced data. Starting values can be difficult to determine, but one strategy is to fit simpler models and use these.

For `asreml`, a single value or a vector of values can be provided, and these are assigned to the argument `init`. For more than one value it is important to match exactly the order on which the parameters are defined. To facilitate this, the option `start.values = TRUE` can be used. Starting values can also be specified using the `vparameters.table` (more details can be found in the following sections).

Next we present some examples for a simple and complex model where we specify starting values.

```
rcb.asr <- asreml(
    fixed = yield ~ Variety,
    random = ~idv(Rep, init = 10),
    residual = ~idv(units, init = 50),
    na.action = na.method(x = "include"),
    data = nin89
)
```

```
initR <- c(60, 0.90, 60, 73, 308, 434, 380)
grass3.asr <- asreml(
    fixed = y ~ Tmt + Time + Tmt:Time,
    residual = ~id(Plant):exph(Time, init = initR),
    data = grassUV
)
```

4.5.2 The vcc argument to `asreml()`

The `vcc` argument to `asreml()` allows users to define equality, multiplicative and linear relationships among variance parameters. This is done by supplying a two-column numeric matrix with a `dimnames` attribute to `vcc`. The first column defines the grouping of variance parameters by assigning the same number to each parameter within a group, and the second column contains the scaling coefficients. The `dimnames()[[1]]` attribute must match the component names in the `asreml` parameter vector (see `start.values`). The parameters in a group are scaled relative to the first parameter in that group so that the scaling of the first parameter in each group is 1.

For example, consider a MET with two tests (under factor `Trial` with levels `1` and `2`) with separate error variances (σ_1^2 and σ_2^2) and the spatial row (ρ_{r_1} and ρ_{r_2}) and column (ρ_{c_1} and ρ_{c_2}) parameters for a separable autoregressive spatial model of order 1 for each trial. Say we wish to constrain these error models to be equal so that $\sigma_1^2 = \sigma_2^2$, $\rho_{r_1} = \rho_{r_2}$ and $\rho_{c_1} = \rho_{c_2}$. Then the appropriate `vcc` matrix with row attributes is

$$\begin{array}{l} \texttt{Trial\_1!R} \\ \texttt{Trial\_1!Row!cor} \\ \texttt{Trial\_1!Column!cor} \\ \texttt{Trial\_2!R} \\ \texttt{Trial\_2!Row!cor} \\ \texttt{Trial\_2!Column!cor} \end{array} \begin{bmatrix} 1 & 1 \\ 2 & 1 \\ 3 & 1 \\ 1 & 1 \\ 2 & 1 \\ 3 & 1 \end{bmatrix}$$

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

$$\begin{array}{l} \texttt{Trial\_1!R} \\ \texttt{Trial\_1!Row!cor} \\ \texttt{Trial\_1!Column!cor} \\ \texttt{Trial\_2!R} \\ \texttt{Trial\_2!Row!cor} \\ \texttt{Trial\_2!Column!cor} \end{array} \begin{bmatrix} 1 & 1 \\ 2 & 1 \\ 3 & 1 \\ 1 & 2 \\ 2 & 1 \\ 3 & 1 \end{bmatrix}$$

4.5.3 The vcm argument to `asreml()`

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

To illustrate, consider an experiment in which two or more separate trials are sown adjacent to one another at the same location, with trials sharing a common plot boundary. In this case it might be sensible to fit the same spatial parameters and error variances for each trial. In other situations it can be sensible to define the same variance structure over several model terms. With the argument `vcm()` linear relationships among variance structure parameters can be defined through a simple linear model and by supplying a design matrix for a set of parameters.

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

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

where $\boldsymbol{\kappa}_n$ is the c-vector of parameters in the new set. In the simple case where the r parameters are constrained to be equal, $c = 1$, the r original parameters are all equal to the one new parameter and $\boldsymbol{M}$ will contain a column of ones.

Consider again the MET with two trials in which we wish to constrain the trial error variances and the spatial row and column parameters for a separable autoregressive spatial model of order 1 for each trial, to be equal. In this case the relationship between the original and new parameter sets is: $\boldsymbol{\kappa} = \boldsymbol{M}\boldsymbol{\kappa}_n$ where $\boldsymbol{\kappa}$ is the 6×1 vector $[\sigma^2_1,\ \rho_{r_1},\ \rho_{c_1},\ \sigma^2_2,\ \rho_{r_2},\ \rho_{c_2}]'$, $\boldsymbol{\kappa}_n$ is a 3×1 vector $[\sigma^2_e,\ \rho_r,\ \rho_c]'$ and $\boldsymbol{M}$, with row attributes, is the 6×3 matrix:

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

Generating M

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

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

$$\begin{array}{ll} v_{r,c} = v_{3,c} & (r = 4,5,6,7;\ c = 1,2) \\ v_{r,r} = v_{3,3} & (r = 4,5,6,7) \\ v_{r,c} = v_{4,3} = 0 & (r = 5,6,7;\ c = 3,4,5,6;\ r > c) \end{array}$$

this gives rise to a vector of 7 parameters $\kappa_n = (v_{1,1}, v_{2,1}, v_{2,2}, v_{3,1}, v_{3,2}, v_{3,3}, v_{4,2})'$ and a variance matrix:

$$\left[\begin{array}{ccccccc} v_{1,1} & & & & & & \\ v_{2,1} & v_{2,2} & & & & & \\ v_{3,1} & v_{3,2} & v_{3,3} & & & & \\ v_{3,1} & v_{3,2} & v_{4,3} & v_{3,3} & & & \\ v_{3,1} & v_{3,2} & v_{4,3} & v_{4,3} & v_{3,3} & & \\ v_{3,1} & v_{3,2} & v_{4,3} & v_{4,3} & v_{4,3} & v_{3,3} & \\ v_{3,1} & v_{3,2} & v_{4,3} & v_{4,3} & v_{4,3} & v_{4,3} & v_{3,3} \end{array}\right]$$

That is, there are only 7 distinct parameters from the original 28 and one of these is to be fixed at 0. Furthermore, suppose that none of the remaining variance parameters from other terms in the model are to be subject to any constraints.

The following call

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

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

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

```
gam <- model.gam$vparameters.table[47:74,]
gam$fac <- factor(c(
      1,
      2,3,
      4,5,6,
      4,5,7,6,
      4,5,7,7,6,
      4,5,7,7,7,6,
      4,5,7,7,7,7,6))

M <- model.matrix(~ -1 + fac, data = gam)
dimnames(M)[[1]] <- row.names(model.gam$vparameters.table)[47:74]
attr(M, "assign") <- NULL
attr(M, "contrasts") <- NULL
```

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

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

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

The final step before fitting the model is to fix the parameters corresponding to level 7 of `gam$fac` to 0. This is achieved by by setting the appropriate values in the `Value` field of `gam` to zero and the corresponding boundary constraint codes in the `Constraint` field to `F`. During the estimation procedure the new parameters, κ_n, use the numbering system of the original parameters, κ, hence the 7 κ_n parameters are numbered from 47-53. So to fix $v_{4,3}$, parameter 53 is fixed.

The modified values and the matrix $\boldsymbol{M}$ are used through the `G.param` and `vcm()` arguments to `asreml()`, that is

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

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

Important Testing has indicated that if the modelled parameters $\boldsymbol{\kappa} = \boldsymbol{M}\boldsymbol{\kappa}_n$ need to be restrained, the variance parameters are not always at their optimum values.

4.6 Estimates from functions of variance components

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

For example, in the data set `coop` we fit a bivariate model for the responses `ywt` and `fat`, and estimate heritability, genetic correlation and phenotypic correlation, as shown below:

```
head(coop, 8)
```

```
     tag sire   dam grp    sex         brr litter age  wwt  ywt gfw fdm fat
1 500001    1 10001  18 Female Twin-Single   2737  31 37.0 48.0 3.2  NA  NA
2 500002    1 10002  18 Female   Twin-Twin   2738  40 28.5 42.5 2.7  NA  NA
3 500003    1 10002  18   Male   Twin-Twin   2738  40 30.0 49.0 2.5  NA  NA
4 500004    1 10003  18   Male Twin-Single   2739  46 44.0 53.5 3.0  NA  NA
5 500005    1 10004  18   Male      Single   2740  54 43.0 59.5 2.5  NA  NA
6 500006    1 10005  18   Male      Single   2741  34 39.5 52.5 3.5  NA  NA
7 500007    1 10006  18 Female      Single   2742  30 43.5 53.5 3.3  NA  NA
8 500008    2 10007  19 Female Twin-Single   2743  54 36.5   NA  NA  NA  NA
```

```
ywt0.sv <- asreml(
    fixed = cbind(ywt, fat) ~ trait + trait:age +
      trait:con(brr) + trait:sex + trait:sex:age,
    random = ~us(trait):id(sire),
    sparse = ~trait:grp,
    residual = ~id(units):us(trait),
    data = coop,
    start.values = TRUE
)
ywt0.sv <- ywt0.sv$vparameters.table
ywt0.sv[, "Value"] <- c(1.4, 0.13, 0.03, 1, 23, 2.5, 1.6)
```

```
ywt0.sv
```

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

```
ywt.asr <- asreml(
    fixed = cbind(ywt, fat) ~ trait + trait:age +
      trait:con(brr) + trait:sex + trait:sex:age,
    random = ~us(trait):id(sire),
    sparse = ~trait:grp,
    residual = ~id(units):us(trait),
    data = coop,
    G.param = ywt0.sv,
    R.param = ywt0.sv
)
```

```
# Variance parameters and their index (number).
cbind.data.frame(
  summary(ywt.asr)$varcomp,
  number = seq_along(ywt.asr$vparameters))
```

```
                           component  std.error   z.ratio bound %ch number
trait:sire!trait_ywt:ywt   1.45821148 0.39814260  3.662536     P   0      1
trait:sire!trait_fat:ywt   0.13027963 0.06792169  1.918086     P   0      2
trait:sire!trait_fat:fat   0.03443794 0.01694845  2.031923     P   0      3
units:trait!R              1.00000000         NA        NA     F   0      4
units:trait!trait_ywt:ywt 23.20554057 0.52214640 44.442594     P   0      5
units:trait!trait_fat:ywt  2.50401740 0.13491632 18.559781     P   0      6
units:trait!trait_fat:fat  1.66291555 0.05066818 32.819725     P   0      7
```

```
# heritA = 4*V1/(V1 + V5)
vpredict(ywt.asr, heritA ~ 4*V1/(V1 + V5))
```

```
        Estimate         SE
heritA 0.2364947 0.06117935
```

```
# heritB = 4*V3/(V3 + V7)
vpredict(ywt.asr, heritB ~ 4*V3/(V3 + V7))
```

```
         Estimate         SE
heritB 0.08115678 0.03936332
```

```
# Genetic correlation.
vpredict(ywt.asr, gc ~ V2/sqrt((V1*V3)))
```

```
    Estimate        SE
gc 0.5813634 0.2038863
```

```
# Phenotypic correlation.
vpredict(ywt.asr, pc ~ (V2+V6)/sqrt(((V1 + V5)*(V3 + V7))))
```

```
    Estimate         SE
pc 0.4071449 0.01832069
```

4.7 Switching between the gamma and sigma parameterization

For single-section models where the residual model can be expressed as a scaled residual matrix, ASReml-R offers the option of fitting parameters using either the sigma parameterization (with sigma parameters, see Section 2.1.1), or the gamma parameterization (with gamma parameters, see Section 2.1.6). Specifying the residual model as a variance structure (or with `dsum()` for multi-section models, Section 3.7) forces ASReml-R to use the sigma parameterization. For example:

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

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

```
met.asr <- asreml(
    fixed = yield ~ at(check, "1"):gen,
    random = ~idv(ibk) + at(check, "0"):gen,
    residual = ~idv(units),
    data = met
)
```

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

```
                  component std.error   z.ratio bound %ch
ibk!ibk            2.116722  1.450487  1.459318     P   0
at(check, 0):gen  29.047878  4.930539  5.891421     P   0
units!units      245.046791  7.059791 34.710203     P   0
units!R            1.000000        NA        NA     F   0
```

Note that in this case $var(\boldsymbol{e}) = \sigma_e^2 \boldsymbol{I}_{2700}$. The overall scale parameter `units!R` is fixed at `1.0` and there is an estimated `units!units` variance. For both correlation (gamma parameterization) and variance (sigma parameterization) models for the residual, ASReml-R automatically includes an overall scale parameter. When a variance model (with associated variance parameter) is specified, the overall scale parameter is fixed at 1.0 to avoid overparameterization. This is reflected by the constraint on `units!R` in the above summary table.

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

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

```
asreml.options(gammaPar = TRUE)
met.asr <- asreml(
    fixed = yield ~ at(check, "1"):gen,
    random = ~idv(ibk) + at(check, "0"):gen,
    residual = ~units,
    data = met
)
```

Note the message indicating that the gamma parameterization has been used. By default, the sigma parameterization is used for reporting parameters:

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

```
                  component std.error   z.ratio bound %ch
ibk!ibk            2.116721  1.450357  1.459448     P   0
at(check, 0):gen  29.047873  4.930575  5.891376     P   0
units!R          245.046732  7.059801 34.710149     P   0
```

But, variance ratios estimated using the gamma parameterization can be reported by setting the `param` argument to `"gamma"` as shown below:

```
                       gamma  component std.error   z.ratio bound %ch
ibk!ibk           0.008638031   2.116721  1.450357  1.459448     P   0
at(check, 0):gen  0.118540135  29.047873  4.930575  5.891376     P   0
units!R           1.000000000 245.046732  7.059801 34.710149     P   0
```

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

The gamma parameterization by default

To ensure upward compatibility with previous releases, `asreml()` also uses the gamma parameterization for model fitting by default if either no residual formula is specified or the residual formula specifies a correlation structure. For example, in the following cases:

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

These will all use the gamma parameterization. The following is the code and default table of variance parameter estimates when an ID variance structure is specified for the residual model:

```
met.asr <- asreml(
    fixed = yield ~ at(check, "1"):gen,
    random = ~idv(ibk) + at(check, "0"):gen,
    residual = ~units,
    data = met
)
```

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

```
                    component std.error   z.ratio bound %ch
ibk!ibk              2.116721  1.450357  1.459448     P   0
at(check, 0):gen    29.047873  4.930575  5.891376     P   0
units!R            245.046732  7.059801 34.710149     P   0
```

Note that a single `units` terms is presented as part of the R structure.

4.8 Comparing models with different variance structures

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

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

Chapter 5

Specifying variance structures using genetic and other relationships

5.1 Introduction

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

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

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

1. The function `ainverse()` takes a pedigree data frame and returns the $\boldsymbol{A}^{-1}$ matrix in sparse form.
2. The matrix from step 1 is included in an `asreml()` analysis using the `vm()` function.

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

5.2 Pedigree and the numerator relationship matrix

5.2.1 Pedigree objects

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

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

We illustrate the use of the pedigree using the data in Harvey (1977). For example, the first 20 lines of `asreml::harvey.ped` are:

```
     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 Genetic groups

If all individuals belong to one genetic group then, as above, use 0 as the identity of the parents of base individuals. However, if base individuals belong to various genetic groups, this can be specified using the `groups` argument to `ainverse()`. The pedigree data frame must then identify these groups. All base individuals should have group identifiers as parents. In this case the identity 0 will only appear in the group identity rows, as in the following example where three sire lines (`G1`, `G2`, an `G3`) are fitted as genetic groups.

```
     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.

5.2.3 Additional qualifiers for numerator relationships

ASReml-R has a number of optional modifications on the calculation of the numerator relationship matrix. All of these can be specified within the function `ainverse()`. Some of these include, elements such as:

- Specifying the level of selfing or inbreeding of an individual.
- Incorporating a column with the gender of an individual. This is relevant to form a relationship matrix for the X chromosome.
- Indicating genetic groups in the pedigree (as shown before).

- Incorporating partial selfing (as a probability).
- Allows for the specification of the inbreeding coefficient for base individuals.
- It can consider the male parent of the female parent (maternal grand-sire) rather than the female parent.

5.3 Specifying relationship and inverse relationship matrices

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

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

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

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

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

```
 [1] "Sire_1" "Sire_2" "Sire_3" "Sire_4" "Sire_5" "Sire_6" "Sire_7" "Sire_8"
 [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"
```

```
head(harvey.ainv)
```

```
     Row Column Ainverse
[1,]   1      1 3.666667
[2,]   2      2 3.666667
[3,]   3      3 2.666667
[4,]   4      4 3.666667
[5,]   5      5 3.333333
[6,]   6      6 3.000000
```

5.4 Generating an A-inverse matrix using `ainverse()`

The function `ainverse()` uses the method of Meuwissen and Luo (1992) to compute the inverse relationship matrix directly from the pedigree. A complete description of the arguments and return value of a call to `ainverse()` is given in the ASReml-R help, available by typing `help(ainverse)`.

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

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

And the variance components are given by:

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

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

and the first five E-BLUP estimates are accessed using:

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

```
                                effect
vm(Calf, harvey.ai)_Sire_1  11.020726
vm(Calf, harvey.ai)_Sire_2 -17.630441
```

```
vm(Calf, harvey.ai)_Sire_3   6.609715
vm(Calf, harvey.ai)_Sire_4  -8.014842
vm(Calf, harvey.ai)_Sire_5   8.014842
vm(Calf, harvey.ai)_Sire_6   8.477982
```

or

```
head(summary(adg0.asr, coef = TRUE)$coef.random)
```

```
                            solution std.error    z.ratio
vm(Calf, harvey.ai)_Sire_1  11.020726  17.43958  0.6319377
vm(Calf, harvey.ai)_Sire_2 -17.630441  17.43958 -1.0109443
vm(Calf, harvey.ai)_Sire_3   6.609715  18.01996  0.3667996
vm(Calf, harvey.ai)_Sire_4  -8.014842  18.75756 -0.4272860
vm(Calf, harvey.ai)_Sire_5   8.014842  18.75756  0.4272860
vm(Calf, harvey.ai)_Sire_6   8.477982  17.12428  0.4950855
```

This is an example of an individual animal model (I) estimating additive variance, σ^2_A, and an animal model residual, σ^2_{eI}. In this example we see the estimate of σ^2_A is 500.32 and the estimate of σ^2_{eI} is 273.60. The resulting variance matrix has terms $\sigma^2_A + \sigma^2_{eI}$, the phenotypic variance, in the diagonal elements. The only non-diagonal terms (1/4) in the relationship matrix occur between calves with the same sire (half-sibs) and so their covariance is $(1/4)\sigma^2_A$.

Because of this relatively simple covariance structure an equivalent sire model can be fitted estimating a sire variance, σ^2_S and a sire model residual, σ^2_{eS}. In this model the phenotypic variance is given by $\sigma^2_S + \sigma^2_{eS}$ and the covariance between half-sibs is σ^2_S. We therefore expect $\sigma^2_A + \sigma^2_{eI} = \sigma^2_S + \sigma^2_{eA}$ and $\sigma^2_A = 4\sigma^2_S$ so that $\sigma^2_{eI} = \sigma^2_{eS} - 3\sigma^2_S$. An alternative and instructive way of deriving the sire model is to consider a reduced animal (RA) model. The RA model was first introduced by Quaas and Pollak (1980) to obtain predicted additive genetic effects without the need to set up equations for all individuals. In this case, the additive genetic effects for animals without offspring u_{Acalf} are written in terms of their parental effects, u_{Adam}, and a Mendelian sampling $u_{Amendcalf}$ term. In our case the calves have no offspring so:

$$u_{Acalf} = 0.5 \times u_{Asire} + 0.5 \times u_{Adam} + u_{Amendcalf}$$

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

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

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

```
adg1.asr <- asreml(
  fixed = y3 ~ Line,
  random = ~Sire,
  residual = ~units,
  data = harvey
)
```

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

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

and the E-BLUP estimates from the first nine sires are:

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

```
               solution std.error     z.ratio
Sire_Sire_1   5.5094593  8.718470  0.63192960
Sire_Sire_2  -8.8136173  8.718470 -1.01091330
Sire_Sire_3   3.3041580  9.008603  0.36677806
Sire_Sire_4  -4.0066901  9.377169 -0.42728140
Sire_Sire_5   4.0066901  9.377169  0.42728140
Sire_Sire_6   4.2381029  8.560945  0.49505085
Sire_Sire_7   0.2220591  8.296533  0.02676529
Sire_Sire_8 -13.4731042  8.415011 -1.60107980
Sire_Sire_9   9.0129422  8.296533  1.08635041
```

Finally, the log-likelihood value of the two models are given by:

```
adg0.asr$loglik
```

```
[1] -238.8305
```

```
adg1.asr$loglik
```

```
[1] -238.8305
```

We see the final `LogLik` is the same in the two models (-238.831) and the E-BLUPs for sires in the individual animal model are twice those in the sire model. The estimate of σ^2_S is 124.89 and the estimate of σ^2_{eS} is 647.86. These convert to give estimates of $\sigma^2_A = 4 \times \sigma^2_S = 4 \times 124.89 = 499.57$

and $\sigma^2_{eI} = \sigma^2_{eS} - 3 \times \sigma^2_S = 647.86 - 3 \times 124.89 = 273.16$ which agree well with the estimates derived from the animal model of $\sigma^2_A = 500.32$ and $\sigma^2_{eI} = 273.60$.

We note that if the sire variance is over-estimated, perhaps by not fitting all the appropriate fixed effects, the estimate of the animal residual variance found from conversion of the sire model estimates might be negative. If an animal model is fitted, ASReml-R does not allow negative residual variances and the estimate will tend to zero. Estimates of the additive variance can be found by constraining the residual variance to a small positive variance. User-specified general inverse matrices are included in an analysis in the same way. Consider an easily verified example where we define a general inverse matrix for `Sire` in the Harvey data as $0.5 \times \boldsymbol{I}_9$ and estimate a scaled sire variance σ^2_{S2}. We expect $\sigma^2_S \boldsymbol{I}_9 = \sigma^2_{S2} \times (0.5)^{-1} \times \boldsymbol{I}_9 = 2 \times \sigma^2_{S2} \boldsymbol{I}_9$ so $\sigma^2_{S2} = \sigma^2_S/2$.

```
sire.giv <- data.frame(row = seq(1, 9), column = seq(1, 9), value = rep(0.5, 9))
attr(sire.giv, "INVERSE") <- TRUE
attr(sire.giv, "rowNames") <- paste("Sire_", seq(1, 9), sep = "")

adg2.asr <- asreml(
    fixed = y3 ~ Line,
    random = ~vm(Sire, sire.giv),
    residual = ~units,
    data = harvey
)
```

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

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

In this case $\sigma^2_{S2} = 62.44$, which is in good agreement with the converted value $0.5 \times \sigma^2_S = 124.89/2 = 62.45$ from the previous analysis.

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

```
sire.mat <- data.frame(row = seq(1, 9), column = seq(1, 9), value = rep(2.0, 9))
sire.mat <- as.matrix(sire.mat)
attr(sire.mat, "rowNames") <- paste("Sire_", seq(1, 9), sep = "")
attr(sire.mat, "INVERSE") <- FALSE

adg3.asr <- asreml(
    fixed = y3 ~ Line,
    random = ~vm(Sire, sire.mat),
    residual = ~units,
    data = harvey
)
summary(adg3.asr)$varcomp
```

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

Most variance matrices are positive definite so every linear combination of effects has a positive variance. However, there are situations in which a variance might be zero, for example, with a dominance model and fully inbred individuals. To illustrate this, we assume we have an extra individual unrelated to the first 9 with zero variance so that the variance matrix has 10 rows and columns with nine diagonal elements being 2.0 and the last being 0. This matrix is positive semi-definite and to allow ASReml-R to continue using Lagrange multipliers, we indicate this by adding as a singularity code PSD. If the relationship matrix is read in as a vector, then we have:

```
ped.vec <- c(
  2,
  0, 2,
  0, 0, 2,
  0, 0, 0, 2,
  0, 0, 0, 0, 2,
  0, 0, 0, 0, 0, 2,
  0, 0, 0, 0, 0, 0, 2,
  0, 0, 0, 0, 0, 0, 0, 2,
  0, 0, 0, 0, 0, 0, 0, 0, 2,
  0, 0, 0, 0, 0, 0, 0, 0, 0, 0)
attr(ped.vec, "rowNames") <- paste("Sire_", seq(1, 10), sep = "")
attr(ped.vec, "INVERSE") <- FALSE

adg4.asr <- asreml(
  fixed = y3 ~ Line,
  random = ~vm(Sire, ped.vec, "PSD"),
  residual = ~units,
  data = harvey
)
```

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

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

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

```
spped.mat  <- data.frame(
  Row =    c(1, 2, 3, 4, 5, 6, 7, 8, 9, 10),
  Column = c(1, 2, 3, 4, 5, 6, 7, 8, 9, 9),
  Value =  c(1, 1, 1, 1, 1, 1, 1, 1, 1, 1))
spped.mat <- as.matrix(spped.mat)
attr(spped.mat,"rowNames") <- paste("Sire_", seq(1, 10), sep = "")
attr(spped.mat,"INVERSE") <- FALSE

adg5.asr <- asreml(
  fixed = y3 ~ Line,
  random = ~vm(Sire, spped.mat, "NSD"),
  data = harvey
)
```

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

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

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

Chapter 6

Prediction from the linear model

6.1 Introduction

Prediction is the process of forming a linear function of the vector of fixed and/or 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 (or predicted margins). 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 prediction methods require as input a data frame of the factor levels and variate values used to fit the model, augmented by new points for which predictions are required. This approach has limitations; for example, it does not lend itself easily to the notion of averaging over particular factors in the model to form predictions.

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

Random factor terms may contribute to predictions in several ways. They may be: 1) evaluated at a given value(s) specified by the user, 2) averaged over, or 3) omitted from the fitted values used to form the prediction. Averaging over the set of random effects gives a prediction specific to the random effects observed. We describe this as a *conditional* prediction. Omitting the term from the model produces a prediction at the population average (zero), that is, substituting the assumed population mean for an unknown random effect. We call this a *marginal* prediction. Note that in any prediction, some terms may be evaluated as conditional and others at marginal values, depending on the aim of prediction.

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

For covariate terms (fixed or random) the associated effect represents the coefficient of a linear trend in the data with respect to the covariate values. These terms should be evaluated at a given value of the covariate, or averaged over several given values. Omission of a covariate from the predictive model is equivalent to predicting at a zero covariate value, which is often inappropriate.

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

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

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

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

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

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

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

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

6.2 Estimating means: the `predict()` method

The `predict` method is detailed in the help by typing `?predict`. 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 randomized block model with random replicate effects is fitted to this data by:

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

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

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

A component named `pvals` is included in the `rcb.pv` object, and the first few lines of predictions are:

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

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

The average standard error of difference between the predicted means is:

```
rcb.pv$avsed
```

```
 overall
4.979075
```

The options `vcov = TRUE` will provide the full variance-covariance matrix of the predicted values.

6.2.1 The prediction process

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

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. In addition, special model terms `mv` and `units` are always ignored.

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

1. Including the variables in the *classify* set and specifying any non-default values at which predictions are to be made by using the `levels` argument.
2. Specifying the averaging set. The default averaging set is those explanatory variables involved in fixed effect model terms that are not in the classifying set. By default, variables that only define random model terms are ignored. The `average` argument allows these variables to be added to the default averaging set.
3. Determining the linear model terms to use in prediction. The default rule is that all model terms based entirely on the classifying and averaging set are used. The `use` and `ignore` arguments allow this default set of model terms to be modified by adding or removing terms, respectively. The `onlyuse` argument explicitly specifies the model terms to use, ignoring all others. The argument `except` explicitly specifies the model terms not to use, including all others. These arguments may implicitly modify the averaging set by including variables defining terms in the predicted model not in the classify set. It is sometimes easier to specify the classify set and the prediction linear model and allow ASReml-R to construct the averaging set.
4. Choosing the weights for forming means over dimensions in the hyper-table. The default is to average over the specified levels but the `average` argument can be used to specify weights to be used in averaging over a factor.

For example, using the wheat data set, we can fit the model:

```
obj.asr <- asreml(
    fixed = yield ~ location + at(check, "1"):gen,
    random = ~at(location):ibk + at(check, "0"):gen,
    residual = ~idv(units),
    data = met
)
```

And the following `predict` method puts `at(check, "0"):gen` in the classify set, `location` in the averaging set and `ibk` in the ignore set. Consequently, the hyper-table is formed from from model

terms `location` and `at(check, "0"):gen` but ignoring all terms in `at(location):ibk`, and then averaging across sites to produce overall test genotype predictions. See the following code and notes.

```
predict(
  object = obj.asr,
  classify = "check:gen", levels = list(check = "0"))$pvals
)
```

```
Notes:
- The predictions are obtained by averaging across the hypertable
  calculated from model terms constructed solely from factors in
  the averaging and classify sets.
- Use 'average' to move ignored factors into the averaging set.
- at(check, 1) is evaluated at 1.00000
- The simple averaging set: location
- The ignored set: ibk

  check        gen predicted.value std.error    status
1     0    Camelot        61.62569  6.774021 Estimable
2     0    Freeman        61.62569  6.774021 Estimable
3     0 GOODSTREAK        61.62569  6.774021 Estimable
4     0    NE16401        56.80434  2.810215 Estimable
5     0    NE16402        75.94455  2.807791 Estimable
6     0    NE16403        58.40314  2.807875 Estimable
```

6.3 Aliasing

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

1. Linear dependencies among model terms due to over-parameterization 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 parameterization 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 `predict()` checks that predictions are of estimable functions in the sense defined by Searle (1971) (p. 160) and are invariant to the constraint method used.

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

6.4 Complicated weighting

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

The factors in the table of weights are specified with the `present` argument and the table of weights with the `prwts` component of the `present` list. There may be a maximum of two independent lists of factors in the `present` list, and, if specified `prwts` applies to the first list only. The order of factors in the tables of weights must correspond to the order in the `present` list with later factors nested within preceding factors. Check the output to ensure that the values in the tables of weights are applied in the correct order.

To illustrate this, we will consider a rather complicated example from a rotation experiment conducted over several years. This particular evaluation 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 different years. Daily live weight gain was assessed between five and eight times in the various years. To obtain a measure of total productivity in terms of sheep live weight, 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(
  object = asr.model, classify = "year:crop:pasture:lime",
  levels = list(year = 1, crop = 1),
  average = list(month = c(56, 55, 56, 53, 57, 63, 0, 0, 0, 0, 0, 0))
)
```

```
predict(
  object = asr.model, 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(
  object = asr.model, 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(
  object = asr.model, 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(
  object = asr.model, 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(
  object = asr.model, 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

We can predict variety means from an RCB analysis of the NIN field trial data using:

```
nin89.asr <- asreml(
  fixed = yield ~ Variety,
  random = ~Rep,
  residual = ~idv(units),
  na.action = na.method(x = "include"),
  data = nin89
)
nin89.pv <- predict(object = nin89.asr, classify = "Variety")
```

Also, we can predict variety means from the NIN field trial data in the presence of the covariate seed weight `sweight` (note this variable was simulated for illustration):

```
nin89$sweight <- with(set.seed(1208), expr = rnorm(n = nrow(nin89), mean = 2, sd = 1))
nin89.asr <- asreml(
  fixed = yield ~ Variety + sweight,
  random = ~Rep,
  residual = ~idv(units),
  na.action = na.method(x = "include"),
  data = nin89
)
nin89.pv <- predict(object = nin89.asr, classify = "Variety")
```

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

Also, variety means from the NIN field trial data at a specified value of the covariate `sweight`:

```
nin89.asr <- asreml(
  fixed = yield ~ Variety + sweight + Rep,
  na.action = na.method(x = "include"),
  data = nin89
)
nin89.pv <- predict(
  object = nin89.asr,
  classify = "Variety:sweight",
  levels = list(sweight = 2)
)
```

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

The following code will predict genotype effects across sites (locations) based on a simple MET analysis:

```
checks <- met[met$check == 1, ]
obj.asr <- asreml(
  fixed = yield ~ gen,
  random = ~location:gen,
  residual = ~dsum(~units|location),
  data = checks
)
pbj.pv <- predict(object = obj.asr, classify = "gen")
```

The above predicts variety means ignoring the `location:gen` term while

```
obj.pv <- predict(
  object = obj.asr,
  classify = "gen",
  average = list(location = c(1/3, 1/3, 1/3, 0, 0, 0, 0, 0))
)
```

This is similar to before, but in this case only the predictions from the first three locations from the hyper-table are averaged and the rest are ignored.

Using the Orange wether trial, we can predict means for each team and trait using:

```
wether.asr <- asreml(
  fixed = cbind(gfw, fdiam) ~ trait + trait:Year,
  random = ~us(trait):Team,
  residual = ~id(units):us(trait),
  data = owt
)
orange.pv <- predict(object = wether.asr, classify = "trait:Team")
```

This code 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.

Chapter 7

Examples

7.1 Introduction

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

7.2 Split-plot design

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

A standard analysis of these data recognizes 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.

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

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

The data can be accessed directly from the object `oats`, and the fields in this data frame are:

```
head(oats)
```

```
  Blocks Nitrogen Subplots    Variety Wplots yield Column Row nrate
1      1  0.6_cwt        1 Marvellous      1   156      1   1   0.6
2      1  0.4_cwt        2 Marvellous      1   118      2   1   0.4
3      1  0.2_cwt        3 Marvellous      1   140      1   2   0.2
4      1    0_cwt        4 Marvellous      1   105      2   2   0.0
5      1    0_cwt        1    Victory      2   111      1   3   0.0
6      1  0.2_cwt        2    Victory      2   130      2   3   0.2
```

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

```
oats.asr <- asreml(
  fixed = yield ~ Variety + Nitrogen + Variety:Nitrogen,
  random = ~Blocks + Blocks:Wplots,
  residual = ~units,
  data = oats
)
```

The variance components are:

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

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

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

```
wald(oats.asr)
```

```
                 Df  Sum of Sq Wald statistic Pr(Chisq)
(Intercept)       1 43443.5179     245.312491 0.0000000
Variety           2   526.0581       2.970492 0.2264466
Nitrogen          3 20020.5000     113.049748 0.0000000
Variety:Nitrogen  6   321.7500       1.816826 0.9357507
residual (MS)    NA   177.0946             NA        NA
```

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 in this process, that is the variance parameters are held constant at the REML estimates obtained from the currently specified fixed model.

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

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

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

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

```
oats.wld$Wald
oats.wld$stratumVariances
```

```
                 Df denDF    F.inc           Pr
(Intercept)       1     5 245.1000 1.931825e-05
Variety           2    10   1.4850 2.723869e-01
Nitrogen          3    45  37.6900 2.457701e-12
Variety:Nitrogen  6    45   0.3028 9.321988e-01

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

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

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

```
oatsV.pv  <- predict(oats.asr, classify = "Variety", sed = TRUE)
oatsN.pv  <- predict(oats.asr, classify = "Nitrogen", sed = TRUE)
oatsVN.pv <- predict(oats.asr, classify = "Variety:Nitrogen", sed = TRUE)
```

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

```
oatsVN.pv$pvals
```

```
       Variety Nitrogen predicted.value std.error    status
1  Golden_rain    0_cwt        80.00000  9.106977 Estimable
2  Golden_rain  0.2_cwt        98.50000  9.106977 Estimable
3  Golden_rain  0.4_cwt       114.66667  9.106977 Estimable
4  Golden_rain  0.6_cwt       124.83333  9.106977 Estimable
5   Marvellous    0_cwt        86.66667  9.106977 Estimable
6   Marvellous  0.2_cwt       108.50000  9.106977 Estimable
7   Marvellous  0.4_cwt       117.16667  9.106977 Estimable
8   Marvellous  0.6_cwt       126.83333  9.106977 Estimable
9      Victory    0_cwt        71.50000  9.106977 Estimable
10     Victory  0.2_cwt        89.66667  9.106977 Estimable
11     Victory  0.4_cwt       110.83333  9.106977 Estimable
12     Victory  0.6_cwt       118.50000  9.106977 Estimable
```

Below we only show the first six rows and columns of the SED matrix.

```
oatsVN.pv$sed[1:6, 1:6]
```

```
6 x 6 Matrix of class "dspMatrix"
         [,1]     [,2]     [,3]     [,4]     [,5]     [,6]
[1,]       NA 7.682954 7.682954 7.682954 9.715025 9.715025
[2,] 7.682954       NA 7.682954 7.682954 9.715025 9.715025
[3,] 7.682954 7.682954       NA 7.682954 9.715025 9.715025
[4,] 7.682954 7.682954 7.682954       NA 9.715025 9.715025
[5,] 9.715025 9.715025 9.715025 9.715025       NA 7.682954
[6,] 9.715025 9.715025 9.715025 9.715025 7.682954       NA
```

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

```
oatsVN.pv$avsed
```

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

Note that the average SED is calculated from the average variance of differences.

7.3 Unbalanced nested design

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

The experiment was reported by Dempster et al. (1984) and was designed to compare the effect of three doses of an experimental compound (control, low and high) on the maternal performance of rats. Thirty female rats (`Dams`) were randomly split into three groups of 10, and each group randomly assigned to the three different doses. All pups in each litter were weighed. The litters differed both in total size and composition of males and females. Thus, the additional covariate `littersize` was included in the analysis. The differential effect of the compound on male and female pups was also of interest. Three litters had to be dropped from the experiment, which meant that one dose had only 7 dams.

The analysis must account for the presence of between-dam variation, and must also recognise the stratification of the experimental units (pups within litters) and the restricted randomisation of the doses to the dams. An indicative ANOVA decomposition for this experiment is given in Table 7.2.

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

Table 7.2: Rat data: ANOVA decomposition

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

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

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

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

```
                  1           6           7           8
LogLik   74.2174175  87.2396114  87.2397915  87.2397915
Sigma2    0.1967003   0.1653687   0.1652991   0.1652991
DF      315.0000000 315.0000000 315.0000000 315.0000000
stepsz    0.5621388   1.0000000   1.0000000   1.0000000
Dam!Dam   0.1000000   0.5827630   0.5867030   0.5866740
units!R   1.0000000   1.0000000   1.0000000   1.0000000
```

The tables of estimated variance parameters and Wald tests are:

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

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

```
ww <- wald(rats.asr, denDF = "default", ssType = "conditional")
```

```
            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 1.190723e-07
Dose         2  23.9   12.1500   11.5100      A 3.147535e-04
Sex          1 299.8   57.9600   57.9600      A 3.512746e-13
Dose:Sex     2 302.1    0.3984    0.3984      B 6.717519e-01
```

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

A plot of residuals vs fitted values is shown in Figure 7.1, and can be obtained with `plot(rats.asr)`.

Before proceeding we note the possibility of several outliers, in particular unit 66. The weight of this female rat, within litter 9 is only 3.68, compared to weights of 7.26 and 6.58 for two other female sibling pups. This weight appears erroneous, but without knowledge of the actual experiment we retain the observation.

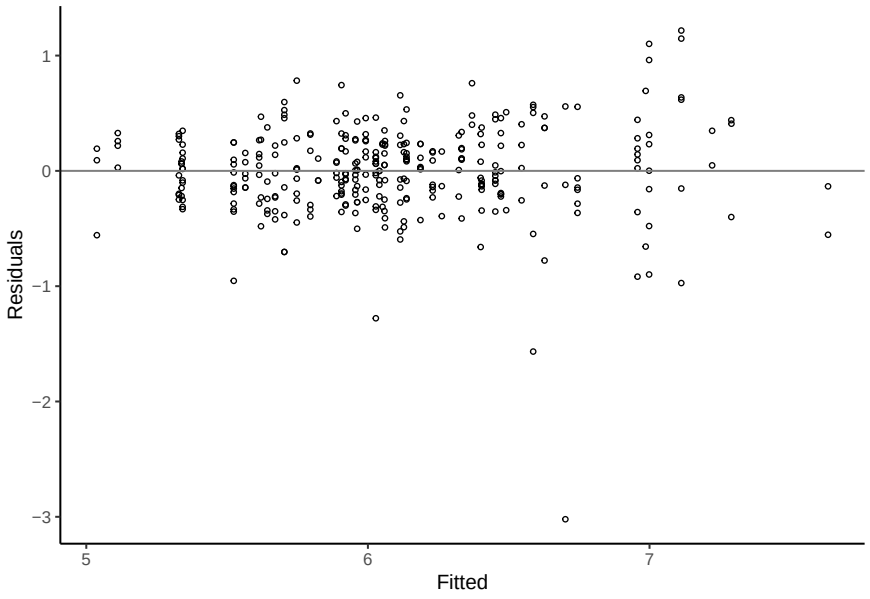


Figure 7.1: Residual plot for the rat data

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

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

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

```
         component  std.error   z.ratio bound %ch
Dam!Dam 0.09791765 0.03341525  2.930328     P   0
units!R 0.16452411 0.01356055 12.132553     P   0
```

```
ww2 <- wald(rats2.asr, denDF = "default", ssType = "conditional")
```

```
            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 1.162738e-07
Sex          1 301.7   59.54   58.27      A 3.032019e-13
Dose         2  24.0   11.42   11.42      A 3.283315e-04
```

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(
  fixed = weight ~ littersize + Dose + Sex,
  residual = ~units,
  data = rats
)
```

```
ww3 <- wald(rats3.asr, denDF = "default", ssType = "conditional")
```

```
            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 1.199158e-06
```

Even if a random term is not *significant*, it should not be dropped from the model if it represents a strata of the design as in this case. The impact of deleting `Dam` on the significance tests for the fixed effects is substantial and not surprising. This reinforces the importance of preserving the strata of the design when assessing the significance of fixed effects. The role of `Dam` in this case is similar to the situation were we need a random effect of *units* to control for pseudo-replication.

7.4 Sources of variability in unbalanced data

This example illustrates an approach to the analysis of unbalanced data where the main aim is to determine the sources of variation rather than assess the significance of imposed treatments. The data are taken from Cox and Snell (1981) and involve an experiment to examine the variability in the production of car voltage regulators. Standard production of regulators involves two steps: 1) regulators are taken from the production line and passed to a *setting station* which adjusts the regulator to operate within a specified range of voltages, and, 2) from the *setting station* the regulator is then passed to a *testing station* where it is tested and returned if outside the required range.

A total of 64 regulators were 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 first few lines of the data set `voltage` are shown below, followed by the `asreml()` function call.

```
  Teststat Setstat Regulatr voltage
1        1       A        1    16.5
2        2       A        1    16.5
3        3       A        1    16.6
4        4       A        1    16.6
5        1       A        2    15.8
6        2       A        2    16.7
```

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.

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

The estimated components of variance are:

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

```
                    component   std.error   z.ratio bound %ch
Teststat         3.287141e-03 0.003337459 0.9849231     P   0
Setstat          1.193738e-02 0.008811857 1.3546953     P   0
Setstat:Teststat 5.175173e-09          NA        NA     B  NA
Setstat:Regulatr 3.077791e-02 0.008452247 3.6413883     P   0
units!R          5.114166e-02 0.005260970 9.7209570     P   0
```

The convergence criterion was satisfied, however, the variance component estimate for the `Setstat:Teststat` term has been fixed at the boundary. The default constraint for variance components is to ensure that the REML estimate remains positive. If an update for any variance component results in a negative value then ASReml-R sets that variance component to a small positive value. If this occurs in subsequent iterations the parameter is fixed at the boundary. The default parameter constraints (Positive for variance components) can be altered (to Unconstrained, 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()`. This is presented in more detail in Section 3.6.1, though it would not generally be recommended for standard analyses.

Below, we produce a residual plot which indicates two unusual data values (Figure 7.2). These values are successive observations, 210 and 211, respectively, being testing stations 2 and 3 for setting station J, regulator 2. These observations will be retained for consistency with other analyses conducted by Cox and Snell (1981).

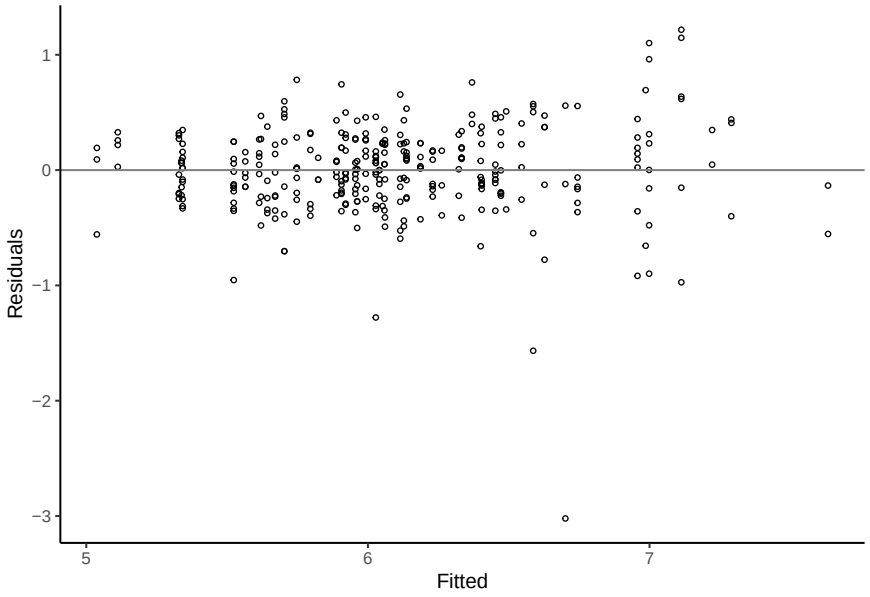


Figure 7.2: Residual vs fitted values for the voltage data

Because the estimated variance component for `Setstat:Teststat` is bound at zero, the model omitting this term returns a REML log-likelihood of 203.242 - the same as the REML log-likelihood for the previous model:

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

A summary of the variance components for the above model is:

```
                  component   std.error   z.ratio bound %ch
Teststat          0.003287145 0.003337467 0.9849221     P   0
Setstat           0.011937344 0.008812595 1.3545776     P   0
Setstat:Regulatr 0.030777971 0.008452070 3.6414712     P   0
units!R           0.051141732 0.005260980 9.7209508     P   0
```

The column labelled `z.ratio` is calculated to give a guide as to the significance of the variance components. This statistic is simply the REML estimate of the variance component divided by its estimated standard error (the square root of the diagonal element 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, given its asymptotic nature. The conclusions using this crude measure are inconsistent with the conclusions obtained from the REML log-likelihood ratio. Further technical details on the REML likelihood ratio test can be found in the Section 2.5.1. For illustration we will be perform a REML likelihood ratio test to compare models `voltage.asr` against `voltage2.asr`.

```
lrt(voltage.asr, voltage2.asr, boundary = TRUE)
```

```
Likelihood ratio test(s) assuming nested random models.
(See Self & Liang, 1987)

                        df LR-statistic Pr(Chisq)
voltage.asr/voltage2.asr  1  -7.2562e-06       0.5
```

Note that the model objects are presented in an increasing order of number of variance components and they have to be nested. Also, the option `boundary = TRUE` specifies that the parameter of interest (in this case the variance component associated with `Setstat:Teststat`) is tested on its parameter space boundary, that is, a positive number.

Further testing for each of the components in the base model is presented in Table 7.3.

Table 7.3: REML log-likelihood ratio test for each variance component in the voltage data

term	log-likelihood	$-2\times$ difference	p-value
`Setstat`	200.31	5.864	0.0077
`Setstat:Regulatr`	184.15	38.19	0.0000
`Ststat:Teststat`	203.24	0.000	0.5000
`Teststat`	199.71	7.064	0.0039

7.5 Balanced repeated measures

The data for this example comes from an experiment conducted at Rothamstead Experimental Station, UK, by J. Lamptey. It consists of a total of five measurements of height (cm) taken on 14 plants. These plants were either diseased or healthy and were arranged in a glasshouse with a completely random design. Plant heights were measured 1, 3, 5, 7 and 10 weeks after the plants were placed in the glasshouse. Therefore, these correspond to repeated measures as each plant was measured five times. There were seven plants in each treatment. The data are illustrated in Figure 7.3.

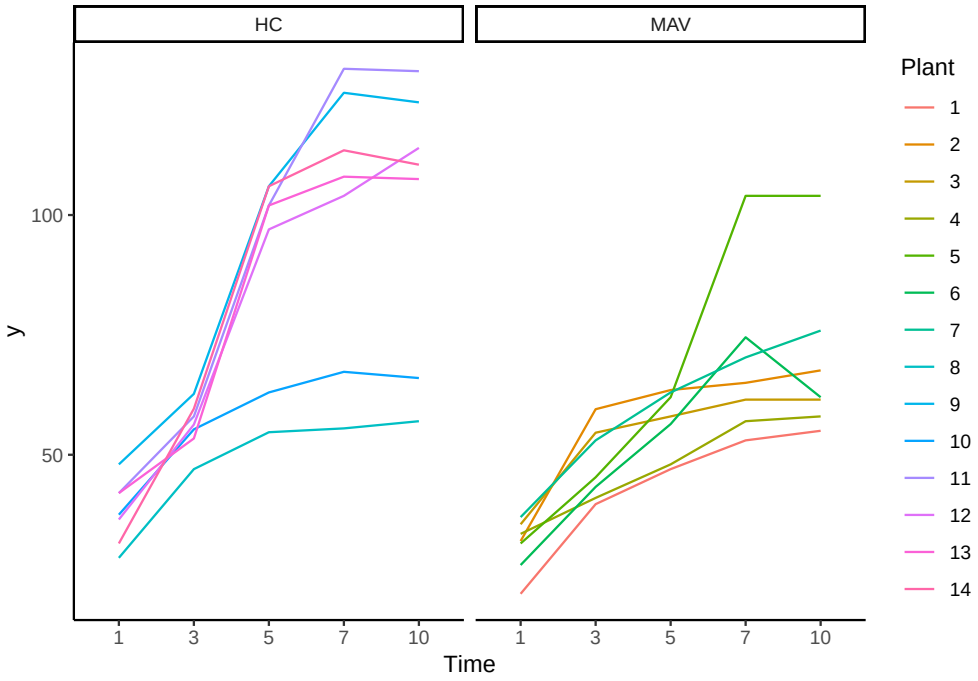


Figure 7.3: Trellis plot of plant height for each of the 14 plants

The following section presents several repeated measures analyses. For some of these it is convenient to arrange the data in a multivariate form, with seven columns containing the plant number, treatment identification and the five height measurements, respectively; while for other analyses, in particular power models, it is necessary to expand the data frame so that the response and a variate for the time of measurement occupy one column each.

The data frame `grass` is in *multivariate* form:

```
  Tmt Plant   y1   y3   y5    y7   y10
1 MAV     1 21.0 39.7 47.0  53.0  55.0
2 MAV     2 32.0 59.5 63.5  65.0  67.6
3 MAV     3 35.5 54.6 58.0  61.5  61.5
4 MAV     4 33.5 41.0 48.0  57.0  58.0
5 MAV     5 31.5 45.3 62.0 104.0 104.0
```

while `grassUV` is in *univariate* form:

```
  Tmt Plant Time HeightID    y
1 MAV     1    1       y1 21.0
2 MAV     1    3       y3 39.7
3 MAV     1    5       y5 47.0
4 MAV     1    7       y7 53.0
5 MAV     1   10      y10 55.0
6 MAV     2    1       y1 32.0
```

The focus is on modelling the error variance for this data. Specifically, we fit the multivariate regression model given by

$$\boldsymbol{Y} = \boldsymbol{DT} + \boldsymbol{E} \tag{7.1}$$

where $\boldsymbol{Y}^{14\times5}$ is the matrix of heights, $\boldsymbol{D}^{14\times2}$ is the trait design matrix, $\boldsymbol{T}^{2\times5}$ is the matrix of fixed effects and $\boldsymbol{E}^{14\times5}$ is the matrix of errors. The heights taken on the same plants will be correlated and so we assume that

$$var(\mathrm{vec}(\boldsymbol{E})) = \boldsymbol{I}^{14} \otimes \boldsymbol{\Sigma} \tag{7.2}$$

where $\boldsymbol{\Sigma}^{5\times5}$ is a symmetric positive definite matrix.

The variance models used in this example for $\boldsymbol{\Sigma}$ are summarized in Table 7.4. They represent some commonly used models for the analysis of repeated measures data (presented by Wolfinger (1996)). Further descriptions of these models can be found in Appendix C. The variance models are fitted by specifying the appropriate special function in the `asreml()` call.

Table 7.4: Summary of variance models fitted to the plant data

model	number of parameters	REML log-likelihood	BIC
uniform	2	-196.88	401.95
power	2	-182.98	374.15
heterogeneous power	6	-171.50	367.57
antedependence (order 1)	9	-160.37	357.51
unstructured	15	-158.04	377.50

The sequence of models given below illustrate some important issues regarding the sort order of the data. In a standard multivariate analysis (data frame `grass`) the response is specified as a matrix and ASReml-R automatically expands the data frame internally to a univariate form in the order **trait nested within units**. The internal factor `units` is created before this expansion. The data frame `grassUV` has been expanded outside ASReml-R in the same order, that is **trait nested**

within experimental units. In this case ASReml-R cannot sensibly create a correct `units` factor so a factor defining the experimental units must already exist - in this case the factor `Plant` can be used.

Note that the sort order of the data must correspond to the order of appearance of the factors in the `residual` formula that defines the experimental units. In the case of the one-dimensional power model, the data must be sorted in the order returned by `unique(x)` where `x` is the column in the data frame containing the distances. In this case ASReml-R checks the sort order and reports an error if incorrect.

Split-plot in time

The split-plot in time model fitting approach can be implemented four ways:

1. By fitting a random `units` term plus an independent residual using the *multivariate* data frame. This model is also known as Uniform.

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

2. By specifying a `corv()` variance model for the R structure, again using the *multivariate* data frame.

```
grass1.asr <- asreml(
  fixed = cbind(y1, y3, y5, y7, y10) ~ trait + Tmt + trait:Tmt,
  residual = ~id(units):corv(trait),
  data = grass
)
```

   As this residual is specified as a correlation matrix the model is fitted on the gamma parameterization. The residual variance is denoted as `S2` in the convergence trace.

3. By fitting `Plant` as a random term plus an independent residual (`Time:Plant`) using the *univariate* data frame.

```
grass1a.asr <- asreml(
  fixed = y ~ Tmt + Time + Tmt:Time,
  random = ~idv(Plant),
  residual = ~id(Plant):idv(Time),
  data = grassUV
)
```

4. By specifying a `corv()` variance model for the `Time:Plant` residual term using the *univariate* data.

```
grass1b.asr <- asreml(
  fixed = y ~ Tmt + Time + Tmt:Time,
  residual = ~id(Plant):corv(Time),
  data = grassUV
)
```

The options **1** and **3** are equivalent as are **2** and **4**. The two forms for Σ are given by

$$\begin{aligned} \boldsymbol{\Sigma} &= \sigma_1^2 \boldsymbol{J} + \sigma_2^2 \boldsymbol{I} \qquad \texttt{idv(units)} \qquad (7.3) \\ \boldsymbol{\Sigma} &= \sigma_e^2 \boldsymbol{I} + \sigma_e^2 \rho (\boldsymbol{J} - \boldsymbol{I}) \qquad \texttt{corv()} \qquad (7.4) \end{aligned}$$

It follows that

$$\begin{aligned} \sigma_e^2 &= \sigma_1^2 + \sigma_2^2 \\ \rho &= \frac{\sigma_1^2}{\sigma_1^2 + \sigma_2^2} \end{aligned} \qquad (7.5)$$

Summaries of the outputs from options **1** and **2** (the `asreml()` calls labelled **Uniform** and **Correlation**, respectively) follow. The REML log-likelihood is the same for both models and it is easy to verify that the REML estimates of the variance parameters satisfy (7.5).

```
summary(grass.asr)$loglik
```

```
[1] -196.8768
```

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

```
                 component std.error  z.ratio bound %ch
units!units        159.8161  75.74758 2.109851     P   0
units:trait!trait  126.4946        NA       NA     F   0
units:trait!R      126.4946  25.82064 4.898972     P   0
```

```
summary(grass1.asr)$loglik
```

```
[1] -196.8768
```

```
summary(grass1.asr)$varcomp
```

```
                        component  std.error  z.ratio bound %ch
units:trait!R         286.3088841 78.3441796 3.654501     P   0
units:trait!trait!cor   0.5581911  0.1303821 4.281196     U   0
units:trait!trait!var 286.3088841         NA       NA     F   0
```

Decaying correlation structure

A more plausible model for repeated measures data would allow the correlations to decrease as the lag increases. The simplest model that accommodates this is the first order autoregressive model. However, since the heights are not measured at equally-spaced time points we use the exponential or power model `exp()`. The correlation function is given by:

$$\rho(u) = \phi^{|u|}$$

where u is the time lag (in this case weeks). The code for this model (also known as Power) is presented below.

```
grass2.asr <- asreml(
  fixed = y ~ Tmt + Time + Tmt:Time,
  residual = ~id(Plant):expv(Time),
  data = grassUV
)
```

The variance parameters from this model are:

```
summary(grass2.asr)$varcomp
```

```
                       component   std.error   z.ratio bound %ch
Plant:Time!R         300.9211889 96.38923337  3.121938     P   0
Plant:Time!Time!pow    0.9190407  0.03120443 29.452250     U   0
Plant:Time!Time!var  300.9211889          NA        NA     F   0
```

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

The code below creates a trend plot (Figure 7.4) of residuals against the factors that index the experimental units.

```
tmp.data <- cbind.data.frame(grassUV, e = grass2.asr$residuals)
ggplot2::ggplot(data = tmp.data, ggplot2::aes(x = Plant, y = e)) +
  ggplot2::facet_wrap(~Time, nrow = 3) +
  ggplot2::geom_point(shape = 1) +
  ggplot2::theme_classic()
```

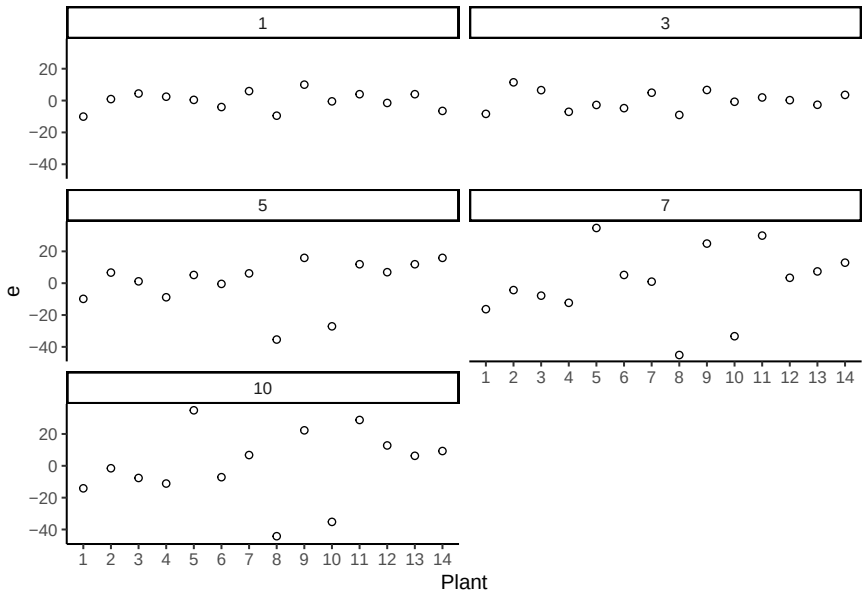


Figure 7.4: Residuals by Plant | Time for the `exp()` variance model of the plant data

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

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

where $\boldsymbol{D}$ is a diagonal matrix of variances and $\boldsymbol{C}$ is a correlation matrix with elements given by $c_{ij} = \phi^{|t_i - t_j|}$.

The code and parameter estimates for this **Heterogeneous power** model are:

```
grass3.asr <- asreml(
  fixed = y ~ Tmt + Time + Tmt:Time,
  residual = ~id(Plant):exph(Time),
  data = grassUV
)
```

```
summary(grass3.asr)$varcomp
```

```
                      component    std.error   z.ratio bound %ch
Plant:Time!R         60.5014787  28.31639025  2.136624     P 0.0
Plant:Time!Time!pow   0.9066574   0.04160873 21.790076     U 0.0
Plant:Time!Time_1    60.5014787           NA        NA     F 0.0
Plant:Time!Time_3    73.1552801  36.95125591  1.979778     P 0.4
Plant:Time!Time_5   307.5585703 137.94251947  2.229614     P 0.1
Plant:Time!Time_7   433.7236072 171.57165789  2.527944     P 0.0
Plant:Time!Time_10  380.2673128 138.70029583  2.741647     P 0.2
```

Note that ASReml-R fixes the scale parameter to 1.0 to ensure that the elements of $\boldsymbol{D}$ are identifiable.

Unstructured correlation structure

The final two models considered are the antedependence model of order 1 and the unstructured model. They consider by default as starting values the gammas of 0.15 for variances and 0.10 for covariances and scales these by 1/2 of the simple variance of the response. This is adequate in many cases (including this example) but we would generally recommend using the REML estimate of $\boldsymbol{\Sigma}$ from a previous model. For example, suitable starting values could be generated from the heterogeneous power model (`grass3.asr`) by:

```
r <- matrix(resid(grass3.asr), nrow = 14, byrow = TRUE)
vcov <- (t(r) %*% r)/12
```

where 12 is the degrees of freedom in this case.

The antedependence form models $\boldsymbol{\Sigma}$ by the inverse Cholesky decomposition

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

where $\boldsymbol{D}$ is a diagonal matrix, and $\boldsymbol{U}$ is a unit upper triangular matrix. For an antedependence model of order q, then $l_{ij} = 0$ for $j > i + q - 1$. The antedependence model of order 1 has nine parameters for these data, five in $\boldsymbol{D}$ and four in $\boldsymbol{U}$. The call using the default starting values was shown before.

The code and antedependence parameter estimates follow and appear successively for each time, that is, the element of $\boldsymbol{D}$ and then the row of $\boldsymbol{U}$:

```
grass4.asr <- asreml(
  fixed = cbind(y1, y3, y5, y7, y10) ~ trait + Tmt + trait:Tmt,
  residual = ~id(units):ante(trait, 1),
  data = grass
)
```

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

```
                            component   std.error    z.ratio bound %ch
units:trait!R              1.000000000          NA         NA     F 0.0
units:trait!trait_y1:y1    0.026866609 0.011023605   2.437189     U 0.0
units:trait!trait_y3:y1   -0.628374025 0.246035667  -2.553996     U 0.0
units:trait!trait_y3:y3    0.037282432 0.015467375   2.410392     U 0.0
units:trait!trait_y5:y3   -1.491096654 0.586492845  -2.542395     U 0.1
units:trait!trait_y5:y5    0.005996185 0.002467878   2.429692     U 0.0
units:trait!trait_y7:y5   -1.280576604 0.206798510  -6.192388     U 0.0
units:trait!trait_y7:y7    0.007896552 0.003232983   2.442497     U 0.0
units:trait!trait_y10:y7  -0.967807268 0.062828991 -15.403833     U 0.0
units:trait!trait_y10:y10  0.039063461 0.015947580   2.449491     U 0.0
```

In this particular case the residual specifies a variance matrix so the sigma parameterization is used for estimation (see Section 2.1.1). The default for multivariate analyses is to use the sigma parameterization.

Finally, the code and estimated components for the unstructured model using default starting values is

```
asreml.options(gammaPar = FALSE, ai.sing = TRUE)
grass5.asr <- asreml(
  fixed = cbind(y1, y3, y5, y7, y10) ~ trait + Tmt + trait:Tmt,
  residual = ~id(units):us(trait),
  data = grass
)
```

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

```
                          component std.error  z.ratio bound %ch
units:trait!R               1.00000        NA       NA     F   0
units:trait!trait_y1:y1    37.22619  15.19746 2.449501     P   0
units:trait!trait_y3:y1    23.39345  13.20622 1.771397     P   0
units:trait!trait_y3:y3    41.51952  16.95020 2.449500     P   0
units:trait!trait_y5:y1    51.65238  32.03385 1.612431     P   0
units:trait!trait_y5:y3    61.91690  34.87146 1.775575     P   0
units:trait!trait_y5:y5   259.12143 105.78573 2.449493     P   0
units:trait!trait_y7:y1    70.81131  46.13796 1.534773     P   0
units:trait!trait_y7:y3    57.61452  46.74179 1.232613     P   0
units:trait!trait_y7:y5   331.80679 145.20148 2.285147     P   0
units:trait!trait_y7:y7   551.50690 225.15083 2.449500     P   0
units:trait!trait_y10:y1   73.78571  46.21260 1.596658     P   0
units:trait!trait_y10:y3   62.56905  46.92685 1.333331     P   0
units:trait!trait_y10:y5  330.85060 144.32316 2.292429     P   0
units:trait!trait_y10:y7  533.75583 220.58712 2.419705     P   0
units:trait!trait_y10:y10 542.17548 221.34133 2.449499     P   0
```

The antedependence model of order 1 is clearly the more parsimonious model (see Table 7.4). There is a surprising level of discrepancy between models for the Wald tests (see Table 7.5 below). The main effect of treatment is significant for the uniform and power models. In this case, we have used `wald(..., denDF = "numeric", ssType = "conditional")`.

7.6 Spatial analysis of a field experiment

This section illustrates spatial and incomplete block analyses of a field experiment using ASReml-R. There has been a large amount of interest in developing techniques for the analysis of spatial data both in the context of field experiments and geostatistical data (Cressie 1991; Cullis and Gleeson

Table 7.5: Summary of Wald statistics for fixed effects for the models fitted to the plant data

model	`Tmt` (df = 1)	p-value	`trait:Tmt` (df = 4)	p-value
uniform	9.41	0.0098	5.10	0.0017
power	6.87	0.0229	6.12	0.0004
heterogeneous power	0.00	0.9836	4.32	0.0105
antedependence (order 1)	4.14	0.0644	3.35	0.0384
unstructured	1.72	0.2149	3.34	0.0612

1991; Gilmour et al. 1997, to name a few). This example illustrates the analysis of *so-called* regular spatial data, in which the data are observed on a lattice or regular grid. This is typical of most small-plot designed field experiments. Spatial data are often irregularly spaced, either by design or because of the observational nature of the study. The techniques we present here 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 used here 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 six replicates laid out in a 10×15 rectangular grid. Table 7.6 shows the layout of the experiment and the coding of the replicates and lattice blocks. The first few rows in the data frame `shf` are:

```
  Rep RowBlk ColBlk Row Column Variety yield
1   1      1      1   1      1       1  1003
2   1      1      2   1      2       2  1356
3   1      1      3   1      3       4  1412
4   1      1      4   1      4       3  1239
5   1      1      5   1      5       5  1508
6   2     11      6   1      6      19  1967
```

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` help to identify the spatial layout of the plots.

Three models are considered below: two spatial and the traditional incomplete block design (for comparative purposes). In the first model, we fit a separable first order autoregressive process to the variance structure of the plot errors. Gilmour et al. (1997) suggest this is often a useful model to commence the spatial modelling process. The form of the variance matrix for the plot errors ($\boldsymbol{R}$-structure) is given by

$$\boldsymbol{\Sigma} = \sigma^2(\boldsymbol{\Sigma}_c \otimes \boldsymbol{\Sigma}_r) \tag{7.6}$$

Table 7.6: Field layout of Slate Hall Farm experiment

Column - Replicate levels															
Row	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	1	1	1	1	1	2	2	2	2	2	3	3	3	3	3
2	1	1	1	1	1	2	2	2	2	2	3	3	3	3	3
3	1	1	1	1	1	2	2	2	2	2	3	3	3	3	3
4	1	1	1	1	1	2	2	2	2	2	3	3	3	3	3
5	1	1	1	1	1	2	2	2	2	2	3	3	3	3	3
6	4	4	4	4	4	5	5	5	5	5	6	6	6	6	6
7	4	4	4	4	4	5	5	5	5	5	6	6	6	6	6
8	4	4	4	4	4	5	5	5	5	5	6	6	6	6	6
9	4	4	4	4	4	5	5	5	5	5	6	6	6	6	6
10	4	4	4	4	4	5	5	5	5	5	6	6	6	6	6
Column - Rowblk levels															
Row	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	1	1	1	1	1	11	11	11	11	11	21	21	21	21	21
2	2	2	2	2	2	12	12	12	12	12	22	22	22	22	22
3	3	3	3	3	3	13	13	13	13	13	23	23	23	23	23
4	4	4	4	4	4	14	14	14	14	14	24	24	24	24	24
5	5	5	5	5	5	15	15	15	15	15	25	25	25	25	25
6	6	6	6	6	6	16	16	16	16	16	26	26	26	26	26
7	7	7	7	7	7	17	17	17	17	17	27	27	27	27	27
8	8	8	8	8	8	18	18	18	18	18	28	28	28	28	28
9	9	9	9	9	9	19	19	19	19	19	29	29	29	29	29
10	10	10	10	10	10	20	20	20	20	20	30	30	30	30	30
Column - Colblk levels															
Row	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
2	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
3	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
4	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
5	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
6	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
7	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
8	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
9	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
10	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30

where the $\mathbf{\Sigma}_c$ and $\mathbf{\Sigma}_r$ are 15×15 and 10×10 correlation matrix functions of the column (ϕ_c) and row (ϕ_r) autoregressive parameters respectively.

The separable autoregressive error model is fitted by:

```
barley1.asr <- asreml(
  fixed = yield ~ Variety,
  residual = ~ar1v(Row):ar1(Column),
  data = shf
)
```

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

```
barley1.asr$loglik
```

```
[1] -700.3226
```

```
summary(barley1.asr)$varcomp
```

```
                          component    std.error   z.ratio bound %ch
Row:Column!R           1.000000e+00           NA        NA     F 0.0
Row:Column!Row!cor     4.583760e-01 8.259721e-02  5.549534     U 0.3
Row:Column!Row!var     3.874983e+04 7.728136e+03  5.014124     P 0.2
Row:Column!Column!cor  6.837836e-01 6.328472e-02 10.804878     U 0.0
```

```
ww <- wald(barley1.asr, denDF = "numeric", ssType = "conditional")
```

```
            Df denDF  F.inc  F.con Margin           Pr
(Intercept)  1  12.8 851.00 851.00        4.420908e-13
Variety     24  80.0  13.04  13.04      A 0.000000e+00
```

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 plot is produced by the `varioGram()` and `plot()` methods, and presented in Figure 7.5. Additional technical details are presented in Section 2.5.3

```
plot(varioGram(barley1.asr))
```

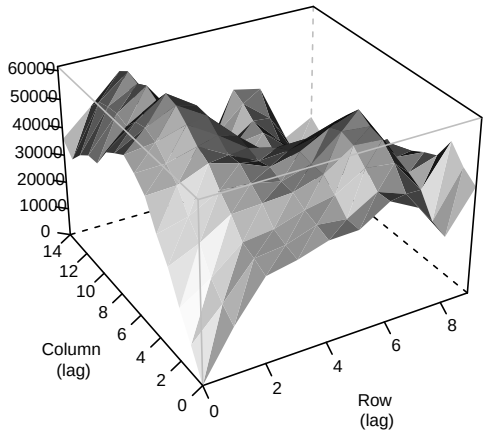


Figure 7.5: Sample variogram of the $AR1$ x $AR1$ model for the Slate Hall data

The iterative sequence has converged to *column* and *row* correlation parameters of 0.6837 and 0.4584, 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 (see 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, it appears in reasonable agreement with the model.

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

```
barley2.asr <- asreml(
  fixed = yield ~ Variety,
  random = ~units,
  residual = ~ar1v(Row):ar1(Column),
  data = shf
)
```

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

$$\boldsymbol{\Sigma} = \sigma^2(\boldsymbol{\Sigma}_c \otimes \boldsymbol{\Sigma}_r) + \psi \boldsymbol{I}_{150} \tag{7.7}$$

where ψ is the ratio of nugget variance to error variance (σ^2). The results show a significant improvement in the REML log-likelihood with the inclusion of the nugget effect (see Table 7.7).

```
barley2.asr$loglik
```

```
[1] -696.8227
```

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

```
                          component    std.error   z.ratio bound %ch
units                 4.861842e+03 1.787924e+03  2.719266     P 0.0
Row:Column!R          1.000000e+00           NA        NA     F 0.0
Row:Column!Row!cor    6.826484e-01 1.022922e-01  6.673516     U 0.0
Row:Column!Row!var    4.579813e+04 1.667622e+04  2.746315     P 0.1
Row:Column!Column!cor 8.437782e-01 6.845720e-02 12.325632     U 0.0
```

```
ww <- wald(barley2.asr, denDF = "numeric", ssType = "conditional")
```

```
            Df denDF  F.inc  F.con Margin          Pr
(Intercept)  1   3.5 259.80 259.80        2.051365e-04
Variety     24  75.7  10.21  10.21      A 2.997602e-15
```

Finally, the incomplete block analysis (with recovery of inter-block information) is:

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

```
barley3.asr$loglik
```

```
[1] -708.1182
```

```
summary(barley3.asr)$varcomp
```

```
        component std.error  z.ratio bound %ch
RowBlk  16777.223  5359.352 3.130457     P 0.1
ColBlk  15884.135  5127.257 3.097979     P 0.0
units!R  8045.005  1336.074 6.021375     P 0.0
```

```
ww <- wald(barley3.asr, denDF = "numeric", ssType = "conditional")
```

```
            Df denDF    F.inc    F.con Margin           Pr
(Intercept)  1  47.3 1893.000 1893.000        0.000000e+00
Variety     24  79.0    8.847    8.847      A 5.950795e-14
```

In general, this variance model is not competitive with the preceding spatial models and this can be noted from the log-likelihood values. The models can be formally compared using the AIC or BIC values, for example. Note that to compare fits using these criteria, the models need to have the same fixed effects (which in this case is only the overall mean), and the smaller the value the better the model.

The Wald statistics for the spatial models (`F-con`) are greater than that for the incomplete block analysis (Table 7.7). We note that the Wald statistic for the spatial model including the nugget effect is smaller than that for the AR1×AR1 model.

Finally, we predict `Variety` means for each model using the `predict()` method. Only the first six 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")
head(barley1.pv$pvals)
```

Table 7.7: Summary of models fitted to the Slate Hall data

model	REML log-likelihood	AIC	BIC	parameters	F-con	SED
AR1×AR1	-700.32	1406.65	1415.13	3	13.04	59.1
AR1×AR1 + units	-696.82	1401.65	1412.96	4	10.21	60.5
incomplete block	-707.79	1423.57	1434.89	4	8.85	62.0

```
  Variety predicted.value std.error    status
1       1        1257.980  64.61798 Estimable
2       2        1501.444  64.98182 Estimable
3       3        1404.987  64.62953 Estimable
4       4        1412.568  64.90618 Estimable
5       5        1514.480  65.59235 Estimable
6       6        1553.458  64.14967 Estimable
```

```
barley1.pv$avsed
```

```
 overall
59.05232
```

```
barley2.pv <- predict(barley2.asr, classify = "Variety")
head(barley2.pv$pvals)
```

```
  Variety predicted.value std.error    status
1       1        1245.582  97.87574 Estimable
2       2        1516.234  97.86387 Estimable
3       3        1403.984  98.25652 Estimable
4       4        1404.918  98.00409 Estimable
5       5        1471.612  98.37730 Estimable
6       6        1521.937  97.98780 Estimable
```

```
barley2.pv$avsed
```

```
 overall
60.51089
```

```
barley3.pv <- predict(barley3.asr, classify = "Variety")
head(barley3.pv$pvals)
```

```
  Variety predicted.value std.error    status
1       1        1284.374  54.68156 Estimable
2       2        1549.437  54.68156 Estimable
3       3        1420.574  54.68156 Estimable
4       4        1452.114  54.68156 Estimable
5       5        1534.198  54.68156 Estimable
6       6        1527.895  54.68156 Estimable
```

```
barley3.pv$avsed
```

```
overall
62.0448
```

7.7 Unreplicated early generation variety trial

This example is a further illustration of the approach to the analysis of field trials presented in the previous section. The data are from an unreplicated field experiment conducted at Tullibigeal in south-western New South Wales, Australia. 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 variety/line every six 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 six replicated commercially available varieties (numbered 527 to 532) were also randomly assigned to plots between three to five plots 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. The first few rows of the data set `wheat` are presented below:

```
  yield weed Column Row Variety
1  2652    0      1   1     526
2  2691    0      2   1     526
3  2770    0      3   1     526
4  2896    0      4   1     526
```

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(
  fixed = yield ~ weed,
  random = ~idv(Variety),
  residual = ~ar1v(Row):id(Column),
  data = wheat
)
```

The REML log-likelihood and variance components from the fit are:

```
wheat1.asr$loglik
```

```
[1] -4239.88
```

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

```
                       component    std.error   z.ratio bound %ch
Variety!Variety    82785.334443 9.220097e+03  8.978792     P 0.1
Row:Column!R           1.000000           NA        NA     F 0.0
Row:Column!Row!cor     0.672194 4.186855e-02 16.054867     U 0.1
Row:Column!Row!var 86295.009311 9.459264e+03  9.122804     P 0.0
```

The REML estimate of the autoregressive parameter indicates substantial within-column spatial correlation.

A two-dimensional spatial model is fitted with:

```
wheat2.asr <- asreml(
  fixed = yield ~ weed,
  random = ~idv(Variety),
  residual = ~ar1v(Row):ar1(Column),
  data = wheat
)
```

```
wheat2.asr$loglik
```

```
[1] -4233.647
```

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

```
                          component    std.error   z.ratio bound %ch
Variety!Variety        8.814084e+04 8.884922e+03  9.920271     P 0.0
Row:Column!R           1.000000e+00           NA        NA     F 0.0
Row:Column!Row!cor     6.854691e-01 4.113764e-02 16.662819     U 0.0
Row:Column!Row!var     8.311660e+04 9.343450e+03  8.895707     P 0.0
Row:Column!Column!cor  2.859709e-01 7.390816e-02  3.869274     U 0.1
```

The change in REML log-likelihood is significant ($\chi^2_1 = 12.46, P < 0.001$) with the inclusion of the autoregressive parameter for `Column`. The sample variogram of the residuals for the $AR1 \times AR1$ model (see Figure 7.6) indicates a linear drift from column 1 to column 10. We will include a linear regression coefficient `lin(Column)` in the model to account for this. But this will not make the REML log-likelihood values comparable as we have different fixed effects.

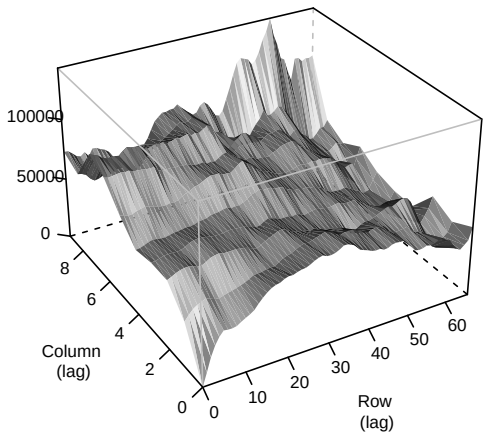


Figure 7.6: Sample variogram of the $AR1$ x $AR1$ model for the Tullibigeal data

```
wheat3.asr <- asreml(
  fixed = yield ~ weed + lin(Column),
  random = ~idv(Variety),
  residual = ~ar1v(Row):ar1(Column),
  data = wheat
)
```

```
wheat3.asr$loglik
```

```
[1] -4227.13
```

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

```
                          component    std.error   z.ratio bound %ch
Variety!Variety        8.897946e+04 8.978980e+03  9.909751     P 0.1
Row:Column!R           1.000000e+00           NA        NA     F 0.0
Row:Column!Row!cor     6.713124e-01 4.290609e-02 15.646088     U 0.1
Row:Column!Row!var     7.780933e+04 8.850037e+03  8.791977     P 0.0
Row:Column!Column!cor 2.660008e-01 7.540594e-02  3.527585     U 0.1
```

```
wald(wheat3.asr, denDF = "numeric", ssType = "conditional")
```

```
            Df denDF    F.inc    F.con Margin           Pr
(Intercept)  1  45.2 7076.000 2193.000        0.000000e+00
weed         1 537.7   91.920   69.980      A 5.551115e-16
lin(Column)  1  50.8    8.732    8.732      A 4.730174e-03
```

```
wheat3.asr$coefficients$fixed
```

```
                effect
(Intercept) 3043.43242
weed        -182.75396
lin(Column)  -31.16488
```

The linear regression of column number on yield is significant (p-value = 0.0047). The sample variogram (Figure 7.7) seems more satisfactory, though interpretation of variograms is often difficult, particularly for unreplicated trials.

The final model includes a nugget effect:

```
wheat4.asr <- asreml(
  fixed = yield ~ lin(Column),
  random = ~idv(Variety) + idv(units),
  sparse = ~weed,
  residual = ~ar1v(Row):ar1(Column),
  data = wheat
)
```

```
wheat4.asr$loglik
```

```
[1] -4221.76
```

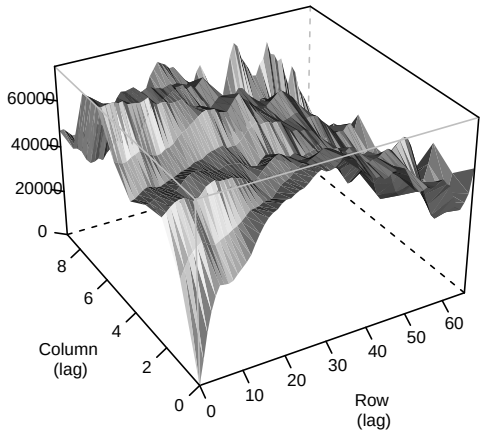


Figure 7.7: Sample variogram of the $AR1$ x $AR1$ + $lin(column)$ model for the Tullibigeal data

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

```
                          component    std.error   z.ratio bound %ch
Variety!Variety        7.376718e+04 1.041730e+04  7.081219     P   0
units!units            3.044589e+04 8.074721e+03  3.770519     P   0
Row:Column!R           1.000000e+00           NA        NA     F   0
Row:Column!Row!cor     8.375155e-01 4.485845e-02 18.670186     U   0
Row:Column!Row!var     5.471356e+04 1.062701e+04  5.148537     P   0
Row:Column!Column!cor  3.753856e-01 1.152838e-01  3.256187     U   0
```

The increase in REML log-likelihood from adding the `units` term is significant. Predicted variety means can be obtained from this model using:

```
wheat4.pv <- predict(
  wheat4.asr,
  classify = "Variety:Column",
  levels = list("Column" = 5.5)
)
```

Note that the predictions are formed at the average value of the variate `Column`, that is, 5.5. Below we only present the first five and last five rows

```
head(wheat4.pv$pvals, 5)
```

```
tail(wheat4.pv$pvals, 5)
```

```
  Variety Column predicted.value std.error    status
1       1    5.5        2915.711  179.3119 Estimable
2       2    5.5        2956.275  178.7804 Estimable
3       3    5.5        2871.298  176.9945 Estimable
4       4    5.5        2985.007  178.7453 Estimable
5       5    5.5        2776.810  179.3485 Estimable

    Variety Column predicted.value std.error    status
528     528    5.5        2725.566  112.2616 Estimable
529     529    5.5        2698.358  103.9254 Estimable
530     530    5.5        3008.925  112.3070 Estimable
531     531    5.5        3018.606  112.2714 Estimable
532     532    5.5        3065.982  112.6677 Estimable
```

It is clear to see 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 examine the matrix of pairwise prediction errors of check varieties. For example, we can use:

```
wheat5.pv <- predict(
  wheat4.asr,
  classify = "Variety:Column",
  levels = list("Variety" = seq(1, 525), "Column" = 5.5)
)
```

and

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

```
wheat6.pv$pvals
```

```
  Variety Column predicted.value std.error    status
1     526    5.5        2384.515  44.21844 Estimable
2     527    5.5        2695.603 133.45777 Estimable
3     528    5.5        2725.566 112.26161 Estimable
4     529    5.5        2698.358 103.92536 Estimable
5     530    5.5        3008.925 112.30699 Estimable
6     531    5.5        3018.606 112.27138 Estimable
7     532    5.5        3065.982 112.66768 Estimable
```

```
wheat6.pv$sed
```

```
7 x 7 Matrix of class "dspMatrix"
          [,1]     [,2]     [,3]      [,4]     [,5]     [,6]     [,7]
[1,]        NA 129.1900 107.2886  98.20994 107.4885 107.0354 107.4494
[2,] 129.19000       NA 165.5509 159.10108 165.5291 165.4921 165.9756
[3,] 107.28860 165.5509       NA 141.34778 148.5680 149.5353 148.2008
[4,]  98.20994 159.1011 141.3478        NA 143.0538 142.1026 143.3975
[5,] 107.48855 165.5291 148.5680 143.05381       NA 144.3516 149.5804
[6,] 107.03544 165.4921 149.5353 142.10261 144.3516       NA 149.5492
[7,] 107.44939 165.9756 148.2008 143.39751 149.5804 149.5492       NA
```

As a final step, and just for completeness, we will present the analyses where we separate the contribution of the check varieties from the test lines by assigning the former to be a fixed effect, and the latter as a random effect. This requires the use of the command `at()` within the model. In the following code we first create the factor `check` with a value of 1 for check varieties and 0 for test lines, and then we fit the corresponding model.

```
wheat$check <- 0
wheat$check[as.numeric(wheat$Variety) >= 526] <- 1
wheat$check <- as.factor(wheat$check)

wheat5.asr <- asreml(
  fixed = yield ~ lin(Column) + at(check, "1"):Variety,
  random = ~at(check, "0"):idv(Variety) + idv(units),
  sparse = ~weed,
  residual = ~ar1v(Row):ar1(Column),
  data = wheat
)
```

Small differences are observed in the estimated variance components between this model and `wheat4.asr` with a moderate increase on the `Variety` component.

```
summary(wheat5.asr)$varcomp
```

```
                                component    std.error   z.ratio bound %ch
at(check, 0):Variety!Variety 7.438597e+04 1.064509e+04  6.987820     P   0
units!units                  2.998101e+04 8.111833e+03  3.695960     P   0
Row:Column!R                 1.000000e+00           NA        NA     F   0
Row:Column!Row!cor           8.359585e-01 4.500441e-02 18.575035     U   0
Row:Column!Row!var           5.529810e+04 1.070748e+04  5.164434     P   0
Row:Column!Column!cor        3.785235e-01 1.145978e-01  3.303059     U   0
```

7.8 Paired case-control study

These data are from an experiment conducted to investigate the tolerance of rice varieties to attack by the larvae of bloodworms. The data have been kindly provided by Dr. Mark Stevens (Yanco Agricultural Institute, Australia). 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 sections present an exhaustive analysis of these data using equivalent univariate and multivariate techniques. It is convenient to use two data frames, one for each approach. The univariate data frame has factors `Pair`, `Run`, `Variety`, `Tmt` and variates `rootwt` and `sqrtroot`. The factor `Pair` labels pairs of trays (to which varieties are allocated) and `Tmt` is the two-level bloodworm treatment factor (control/treated). The first few lines of the univariate data frame, called `rice` are:

```
  Pair   rootwt Run sqrtroot     Tmt  Variety
1    2 313.7220   1 17.71220 Control AliCombo
2    2  86.9457   1  9.32447 Exposed AliCombo
3   24 267.6110   1 16.35880 Control     IR36
4   24  87.1914   1  9.33763 Exposed     IR36
5    3 172.7230   2 13.14240 Control AliCombo
6    3  26.0771   2  5.10657 Exposed AliCombo
```

The multivariate data frame, called `riceMV`, 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), and its first few lines are:

```
  Pair Run   Variety      yc      ye     syc     sye
1    1  60  AliCombo 185.403 70.1814 13.6163 8.37743
2    2   1  AliCombo 313.722 86.9457 17.7122 9.32447
3    3   2  AliCombo 172.723 26.0771 13.1424 5.10657
4    4   3 Bluebelle 193.056 41.9249 13.8944 6.47494
5    5   4 Bluebelle 193.119 68.3390 13.8967 8.26674
6    6   5 Bluebelle 111.228  3.5935 10.5465 1.89565
```

A plot of the treated vs the control root area (on the square root scale) for each variety is shown in Figure 7.8. There is a strong dependence between the treated and control root area, which is not surprising. The aim of the experiment was to determine the tolerance of varieties to bloodworms and identify the most tolerant varieties. The definition of tolerance should allow for the fact that varieties differ in their inherent seedling vigour (as seen in Figure 7.8). The initial approach was to regress the treated root area against the control root area and define the index of vigour as the residual from this regression. This is clearly inefficient since there is error in both variables. We seek to determine an index of tolerance from the joint analysis of treated and control root area.

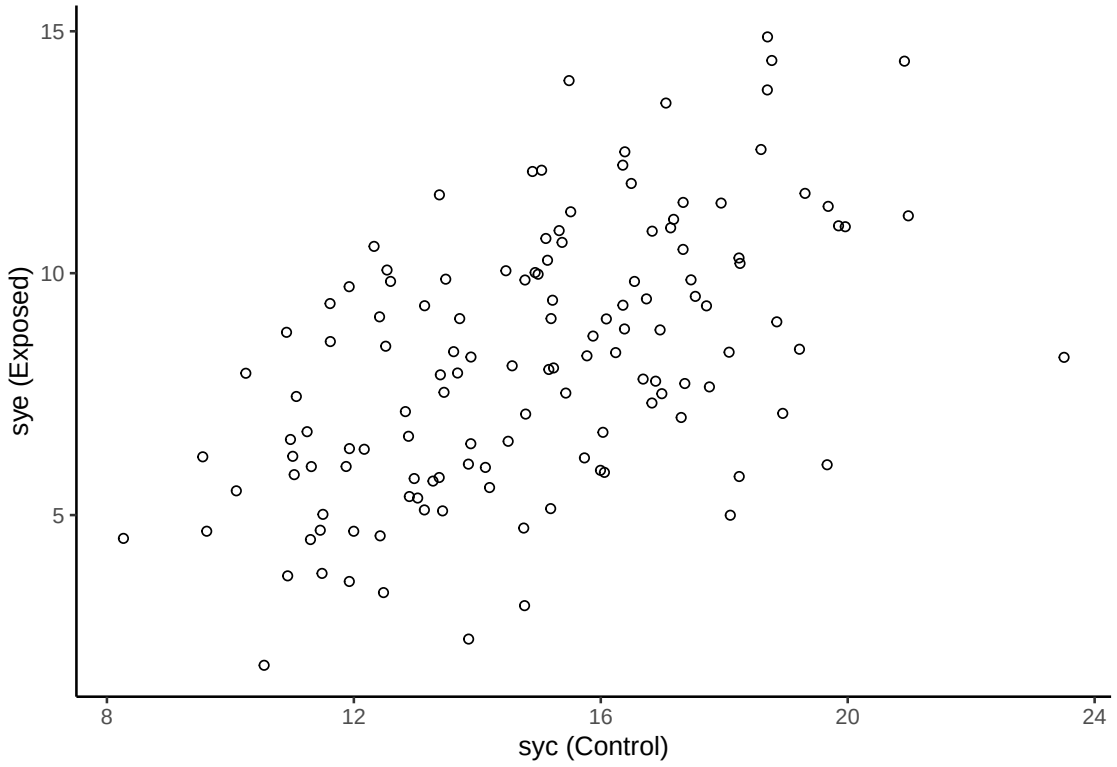


Figure 7.8: Rice bloodworm data: Plot of square root of root weight for treated versus control

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 blocking 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 blocking structure of the form:

```
Run/Tmt/Variety
  = Run + Run:Tmt + Run:Tmt:Variety
  = Run + Run:Tmt + Pair:Tmt
```

Combining the two provides the full blocking 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 `sqrt(rootwt)` but rather two traits, namely `sqrt(rootwt)` in the presence/absence of bloodworms. To minimize selection bias the `Variety` main effects and `Tmt:Variety` interactions are taken as random. The main effect of treatment is fitted as fixed to allow for the likely scenario that rather than a single population of treatment by variety effects there are in fact two populations (control and treated) with a different mean for each. There is evidence of this prior to analysis, with the large difference in mean `sqrt(rootwt)` for the two groups (14.93 and 8.23 for control and treated, respectively). The inclusion of `Tmt` as a fixed effect ensures that E-BLUPs of `Tmt:Variety` effects are shrunk to the correct mean (treatment means rather than an overall mean).

The model for the data are given by

$$\boldsymbol{y} = \boldsymbol{X}\boldsymbol{\tau} + \boldsymbol{Z}_1\boldsymbol{u}_1 + \boldsymbol{Z}_2\boldsymbol{u}_2 + \boldsymbol{Z}_3\boldsymbol{u}_3 + \boldsymbol{Z}_4\boldsymbol{u}_4 + \boldsymbol{Z}_5\boldsymbol{u}_5 + \boldsymbol{e} \tag{7.8}$$

where $\boldsymbol{y}$ is a vector of length $n = 264$ containing the `sqrtroot` values, $\boldsymbol{\tau}$ corresponds to a constant term and the fixed treatment contrast and $\boldsymbol{u}_1 \ldots \boldsymbol{u}_5$ correspond to random effects of: `Variety`, `Tmt:Variety`, `Run`, `Tmt:Run` and `Variety:Run`. The random effects and error are assumed to be independent Gaussian variables with zero means and variance structures: $var(\boldsymbol{u}_i) = \sigma_i^2 \boldsymbol{I}_{b_i}$ (where b_i is the length of $\boldsymbol{u}_i$, $i = 1 \ldots 5$) and $var(\boldsymbol{e}) = \sigma^2 \boldsymbol{I}_n$.

Now, we can proceed to fit the model with the code below, and we request some output.

```
asreml.options(gammaPar = TRUE)
rice1.asr <- asreml(
  fixed = sqrtroot ~ Tmt,
  random = ~idv(Variety) + idv(Variety):id(Tmt) +
            idv(Run) + idv(Pair) + idv(Run):id(Tmt),
  residual = ~units,
  data = rice
)
```

```
summary(rice1.asr)$loglik
summary(rice1.asr)$varcomp
```

```
[1] -345.2559
```

```
                     component std.error   z.ratio bound %ch
Variety!Variety     2.3778037 0.7911195 3.0056189     P 0.2
Run!Run             0.3217226 0.5483794 0.5866789     P 0.9
Variety:Tmt!Variety 0.4923122 0.2764192 1.7810344     P 0.0
Pair!Pair           0.9758347 0.3882604 2.5133509     P 0.1
Run:Tmt!Run         1.7478110 0.4793497 3.6462126     P 0.0
units!R             1.3149769 0.2974428 4.4209405     P 0.0
```

```
ww <- wald(rice1.asr, denDF = "numeric", ssType = "conditional")
```

```
            Df denDF  F.inc  F.con Margin Pr
(Intercept)  1  53.5 1485.0 1485.0         0
Tmt          1  60.4  469.3  469.3      A  0
```

The estimated variance components from this analysis also appear in column (a) of Table 7.8. The variance component for the `Variety` main effects is large. There is evidence of `Variety:Tmt` interactions so we may expect some discrimination between varieties in terms of tolerance to bloodworms.

Given the large difference ($p < 0.001$) between `Tmt` means we may wish to allow for heterogeneity of variance associated with `Tmt`. Thus we fit a separate `Variety:Tmt` variance for each level of `Tmt` so that instead of assuming $var(\boldsymbol{u}_2) = \sigma_2^2 \boldsymbol{I}_{88}$, we assume

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

where σ_{2c}^2 and σ_{2t}^2 are the `Variety:Tmt` interaction variances for control and treated, respectively. This model can be fitted using a diagonal variance structure for the treatment part of the interaction. We also fit a separate `Run:Tmt` variance for each level of `Tmt` and heterogeneity at the residual level,

by including an extra `at(Tmt, 2):units` term. We have chosen level 2 of `Tmt` as we expect more variation for the exposed treatment and thus the extra variance component for this term should be positive.

By default, ASReml-R sets the parameter constraint for variance components to 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 using `start.values = T` in `asreml()`,
- opens the default text editor where the parameter constraints can be changed to `U` and the result saved to `temp.gam`, and
- fits the model using the G level parameter settings in `temp.gam` through the `G.param` argument.

```
asreml.options(gammaPar = TRUE)
temp.gam <- asreml(
  fixed = sqrtroot ~ Tmt,
  random = ~idv(Variety) + id(Variety):diag(Tmt) + idv(Run) + idv(Pair) +
            id(Run):diag(Tmt) + at(Tmt,2):units,
  residual = ~idv(units),
  data = rice,
  start.values = TRUE
)$vparameters.table

temp.gam[c(2, 3), "Constraint"] <- c("U", "U")
temp.gam[c(6, 7), "Constraint"] <- c("U", "U")
temp.gam
```

```
                 Component Value Constraint
1          Variety!Variety   0.1          P
2  Variety:Tmt!Tmt_Control   0.1          U
3  Variety:Tmt!Tmt_Exposed   0.1          U
4                Run!Run     0.1          P
5               Pair!Pair    0.1          P
6      Run:Tmt!Tmt_Control   0.1          U
7      Run:Tmt!Tmt_Exposed   0.1          U
8   at(Tmt, Exposed):units   0.1          P
9                 units!R    1.0          P
10            units!units    1.0          F
```

```
asreml.options(gammaPar = TRUE)
rice2.asr <- asreml(
  fixed = sqrtroot ~ Tmt,
  random = ~idv(Variety) + id(Variety):diag(Tmt) + idv(Run) +
            idv(Pair) + id(Run):diag(Tmt) + at(Tmt,2):idv(units),
  residual = ~idv(units),
  G.param = temp.gam,
  data = rice
)
```

```
summary(rice2.asr)$loglik
summary(rice2.asr)$varcomp
```

```
[1] -343.2199
```

```
                                component std.error    z.ratio bound %ch
Variety!Variety                 2.3341384 0.7761558  3.0073065     P 0.1
Run!Run                         0.3193663 0.5435029  0.5876075     P 0.5
Variety:Tmt!Tmt_Control         1.5055897 0.6644957  2.2657628     U 0.2
Variety:Tmt!Tmt_Exposed        -0.3721460 0.4563015 -0.8155702     U 0.5
Pair!Pair                       0.9875966 0.3811214  2.5912913     P 0.1
Run:Tmt!Tmt_Control             1.3891313 0.6359438  2.1843616     U 0.0
Run:Tmt!Tmt_Exposed             2.2242074 0.7237749  3.0730651     U 0.1
at(Tmt, Exposed):units!units    0.2039961 0.6317102  0.3229267     P 0.3
units!units                     1.1565111        NA         NA     F 0.0
units!R                         1.1565111 0.4173686  2.7709588     P 0.0
```

```
wald(rice2.asr, denDF = "numeric", ssType = "conditional")
```

```
            Df denDF  F.inc  F.con Margin Pr
(Intercept)  1  56.4 1277.0 1277.0         0
Tmt          1  60.6  448.8  448.8      A  0
```

The estimated variance components from this analysis are given in column (b) of Table 7.8. There is no significant variance heterogeneity at the residual or `Run:Tmt` level. This indicates that the square root transformation of the data has successfully stabilized 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 previously) 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,

Table 7.8: Estimated variance components from univariate analyses of bloodworm data: (a) model with homogeneous variance for all terms and (b) model with heterogeneous variance for interactions involving Tmt

source	(a) homogeneous	(b) heterogeneous control	treated
`Variety`	2.378	2.334	
`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	0.988	
`Tmt:Pair`	1.315	1.157	1.360
REML log-likelihood	-345.256	-343.22	

$$var(\mathbf{1}_2 \otimes \boldsymbol{u}_1 + \boldsymbol{u}_2) = \begin{bmatrix} \sigma_1^2 + \sigma_{2c}^2 & \sigma_1^2 \\ \sigma_1^2 & \sigma_1^2 + \sigma_{2t}^2 \end{bmatrix} \otimes \boldsymbol{I}_{44} \tag{7.9}$$

Using the estimates from Table 7.8 this structure is estimated as

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

Thus the variance of the variety effects in the control group (also known as the genetic variance for this group) is 3.84. The genetic variance for the treated group is much lower (1.96). The genetic correlation is $2.33/\sqrt{3.84 \times 1.96} = 0.85$ which is strong, supporting earlier indications of the dependence between the treated and control root area (Figure 7.8).

A multivariate approach

In this simple case in which the variance heterogeneity is associated with the two-level factor `Tmt`, the analysis is equivalent to a bivariate analysis in which the two traits correspond to the two levels of `Tmt`, namely `sqrtroot` for control and treated. The model for each trait is given by

$$\boldsymbol{y}_j = \boldsymbol{X}\boldsymbol{\tau}_j + \boldsymbol{Z}_v\boldsymbol{u}_{v_j} + \boldsymbol{Z}_r\boldsymbol{u}_{r_j} + \boldsymbol{e}_j \quad (j = c, t) \tag{7.10}$$

where $\boldsymbol{y}_j$ is a vector of length $n = 132$ containing the `sqrtroot` values for variate j ($j = c$ for control and $j = t$ for treated), $\boldsymbol{\tau}_j$ corresponds to a constant term by trait and $\boldsymbol{u}_{v_j}$ and $\boldsymbol{u}_{r_j}$ correspond to random variety and run effects. The design matrices are the same for both traits. The random effects and error are assumed to be independent Gaussian variables with zero means and variance structures: $var(\boldsymbol{u}_{v_j}) = \sigma_{v_j}^2\boldsymbol{I}_{44}$, $var(\boldsymbol{u}_{r_j}) = \sigma_{r_j}^2\boldsymbol{I}_{66}$ and $var(\boldsymbol{e}_j) = \sigma_j^2\boldsymbol{I}_{132}$. The bivariate model can be written as a direct extension of (7.10), namely

$$\boldsymbol{y} = (\boldsymbol{I}_2 \otimes \boldsymbol{X})\,\boldsymbol{\tau} + (\boldsymbol{I}_2 \otimes \boldsymbol{Z}_v)\,\boldsymbol{u}_v + (\boldsymbol{I}_2 \otimes \boldsymbol{Z}_r)\,\boldsymbol{u}_r + \boldsymbol{e}^* \tag{7.11}$$

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

There is an equivalence between the effects in this bivariate model and the univariate model of (7.8). The variety effects for each trait ($\boldsymbol{u}_v$ in the bivariate model) are partitioned in (7.8) into variety main effects and `tmt.variety` interactions, so that $\boldsymbol{u}_v = \boldsymbol{1}_2 \otimes \boldsymbol{u}_1 + \boldsymbol{u}_2$. There is a similar partitioning for the run effects and the errors (Table 7.9).

Table 7.9: Equivalence of random effects in bivariate and univariate analyses

effects	bivariate (model 7.11)	univariate (model 7.8)
`trait:Variety`	$\boldsymbol{u}_v$	$\boldsymbol{1}_2 \otimes \boldsymbol{u}_1 + \boldsymbol{u}_2$
`trait:Run`	$\boldsymbol{u}_r$	$\boldsymbol{1}_2 \otimes \boldsymbol{u}_3 + \boldsymbol{u}_4$
`Pair:trait`	$\boldsymbol{e}^*$	$\boldsymbol{1}_2 \otimes \boldsymbol{u}_5 + \boldsymbol{e}$

In addition to the assumptions in the models for individual traits (7.10), the bivariate analysis involves the assumptions $cov(\boldsymbol{u}_{v_c})\boldsymbol{u}_{v_t}' = \sigma_{v_{ct}}\boldsymbol{I}_{44}$, $cov(\boldsymbol{u}_{r_c})\boldsymbol{u}_{r_t}' = \sigma_{r_{ct}}\boldsymbol{I}_{66}$ and $cov(\boldsymbol{e}_c)\boldsymbol{e}_t' = \sigma_{ct}\boldsymbol{I}_{132}$. Thus, random effects and errors are correlated between traits. So, for example, the variance matrix for the variety effects for each trait is given by

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

This unstructured form for `trait:Variety` in the bivariate analysis is equivalent to the `Variety` main effect plus heterogeneous `Variety:Tmt` interaction variance structure (7.9) in the univariate analysis. Similarly, the unstructured form for `trait:Run` is equivalent to the `Run` main effect plus heterogeneous `Run:Tmt` interaction variance structure. The unstructured form for the errors (`Pair:trait`) in the bivariate analysis is equivalent to the `Pair` plus heterogeneous error (`Pair:Tmt`) variance in the univariate analysis.

Hence, the ASReml-R call for this bivariate model is:

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

```
summary(riceMV.asr)$loglik
```

```
[1] -343.22
```

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

```
                              component std.error   z.ratio bound %ch
trait:Variety!trait_syc:syc 3.8370236 1.1041043 3.4752367     P 0.3
trait:Variety!trait_sye:syc 2.3317436 0.7741058 3.0121769     P 0.3
trait:Variety!trait_sye:sye 1.9598665 0.7268965 2.6962112     P 0.3
trait:Run!trait_syc:syc     1.7090708 0.6541103 2.6128176     P 0.3
trait:Run!trait_sye:syc     0.3207441 0.5443446 0.5892299     P 1.7
trait:Run!trait_sye:sye     2.5450260 0.7965487 3.1950665     P 0.2
units:trait!R               1.0000000        NA        NA     F 0.0
units:trait!trait_syc:syc   2.1435663 0.4820833 4.4464646     P 0.0
units:trait!trait_sye:syc   0.9870784 0.3809549 2.5910636     P 0.1
units:trait!trait_sye:sye   2.3471096 0.5073875 4.6258716     P 0.0
```

```
wald(rice2.asr, denDF = "numeric", ssType = "conditional")
```

```
            Df denDF  F.inc  F.con Margin Pr
(Intercept)  1  56.4 1277.0 1277.0         0
Tmt          1  60.6  448.8  448.8      A  0
```

The resultant REML log-likelihood is identical to that of the heterogeneous univariate analysis (column (b) of Table 7.8). The estimated variance parameters are summarized in Table 7.10.

Table 7.10: Estimated variance components from bivariate analysis of bloodworm data

source	control variance	treated variance	covariance
us(trait):Variety	3.84	1.96	2.33
us(trait):Run	1.71	2.55	0.32
Pair:us(trait)	2.14	2.35	0.99

The predicted variety means are obtained using:

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

and the first few rows are:

```
  trait   Variety predicted.value std.error    status
1   syc  AliCombo       14.953221 0.9180989 Estimable
2   syc    Amaroo        7.994086 0.7993077 Estimable
3   syc   Balilla       16.161246 0.9181010 Estimable
4   syc Bluebelle        8.481302 0.7992850 Estimable
5   syc     Bogan       14.420180 0.9185728 Estimable
6   syc       C22        8.230838 0.7995129 Estimable
```

These predictions are on the square root scale; it is straightforward to *back-transform* the predicted means to the original scale of measurement. Approximate standard errors on the original scale can be calculated from a Taylor series approximation (*i.e.*, delta method). That is, if x is a random variable with $E(x) = \theta$, and $y = g(x)$ is some function of x, then $var(y) = (dy/dx)^2_\theta var(x)$ (see Kendall and Stuart (1969) pp 231, for an example). In this case, $g(x) = x^2$ and $g'(x) = dy/dx = 2x$. The following code calculates the transformed predictions and approximate standard errors:

```
pv <- riceMV.pv$pvals
pv$rootwt <- pv$predicted.value^2
pv$approxSE <- sqrt(4 * pv$predicted.value^2 * pv$std.error^2)
pv$est.status <- NULL
```

```
  trait   Variety predicted.value std.error    status    rootwt approxSE
1   syc  AliCombo       14.953221 0.9180989 Estimable 223.59883 27.45707
2   syc    Amaroo        7.994086 0.7993077 Estimable  63.90540 12.77947
3   syc   Balilla       16.161246 0.9181010 Estimable 261.18589 29.67531
4   syc Bluebelle        8.481302 0.7992850 Estimable  71.93248 13.55796
5   syc     Bogan       14.420180 0.9185728 Estimable 207.94158 26.49197
6   syc       C22        8.230838 0.7995129 Estimable  67.74670 13.16132
```

Interpretation of results

Recall that the primary interest is varietal tolerance to bloodworms, and this could be defined in various ways. One option is to consider the regression implicit in the variance structure for the trait by variety effects. The variance structure can arise from a regression of treated variety effects on control effects, namely: $\boldsymbol{u}_{v_t} = \beta \boldsymbol{u}_{v_c} + \boldsymbol{\epsilon}$, where the slope is $\beta = \sigma_{v_{ct}}/\sigma^2_{v_c}$.

Tolerance can be defined in terms of the deviations from regression, $\boldsymbol{\epsilon}$. Varieties with large positive deviations have the greatest tolerance to bloodworms. Note that this is similar to the original approach except that the regression has been conducted at the genotypic rather than the phenotypic level. In Figure 7.9, the E-BLUPs for treated have been plotted against the E-BLUPs for control for each variety and the fitted regression line (slope = 0.61) has been drawn. Varieties with large positive deviations from the regression line include YRK3, Calrose, HR19 and WC1403.

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

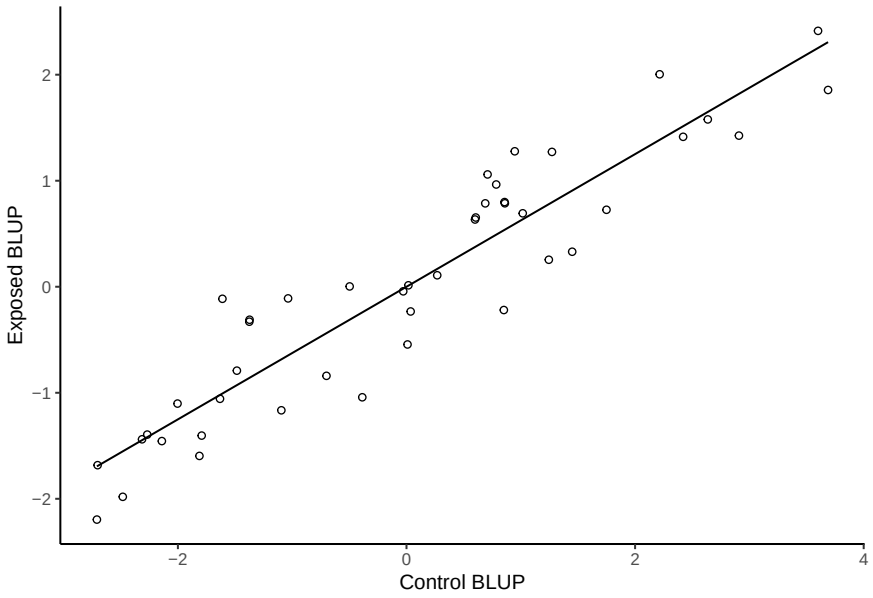


Figure 7.9: E-BLUPs for treated plotted against E-BLUPs for control

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

whereas

$$\begin{aligned} cov(\boldsymbol{\delta})\boldsymbol{u}'_{v_c} &= cov(\boldsymbol{u}_{v_c} - \boldsymbol{u}_{v_t})\boldsymbol{u}'_{v_c} \\ &= \left(\sigma^2_{v_c} - \sigma_{v_{ct}}\right)\boldsymbol{I}_{44} \end{aligned}$$

The independence of $\boldsymbol{\epsilon}$ and $\boldsymbol{u}_{v_c}$ and dependence between $\boldsymbol{\delta}$ and $\boldsymbol{u}_{v_c}$ is clearly illustrated in Figures 7.10 and 7.11. In this example, the two measures have provided very different rankings of the varieties. The choice of tolerance measure depends on the aim of the experiment. Here, the aim was to identify tolerance which is independent of inherent vigour so the method of deviations from regression is preferred.

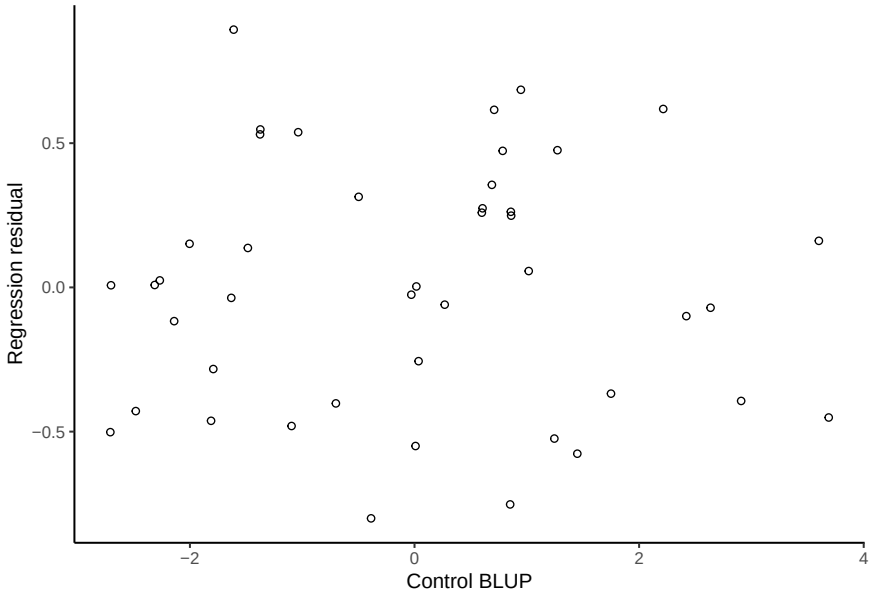


Figure 7.10: Estimated deviations from regression of treated on control for each variety plotted against estimate for control

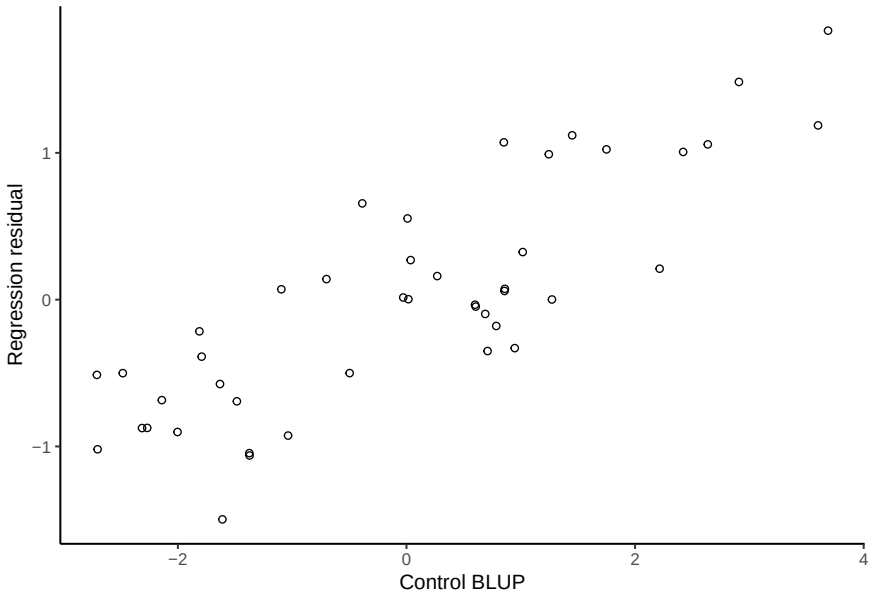


Figure 7.11: Estimated difference between control and treated for each variety plotted against estimate for control

7.9 Balanced longitudinal data - random coefficients and cubic smoothing splines

In this section we illustrate the use of random coefficients and cubic smoothing splines for the analysis of balanced longitudinal data. The implementation of cubic smoothing splines in ASReml-R is based on the mixed model formulation of Verbyla et al. (1999). 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.

The data set used in this illustration was originally presented by Draper and Smith (1998) and has recently been re-analysed by Pinheiro and Bates (2000). The data corresponds to trunk circumferences (in millimetres) of each of five trees taken at seven measurement times (Figure 7.12). All trees were measured at the same time so the data set is 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, defined as:

$$y = \frac{\phi_1}{1 + \exp\left[-(x - \phi_2)/\phi_3\right]} \tag{7.12}$$

where y is the trunk circumference, x is the tree age (in days since December 31 1968), ϕ_1 is the asymptotic height, ϕ_2 is the inflection point or the time at which the tree reaches $0.5\phi_1$, and ϕ_3 is the time elapsed between trees reaching half and about 3/4 of ϕ_1.

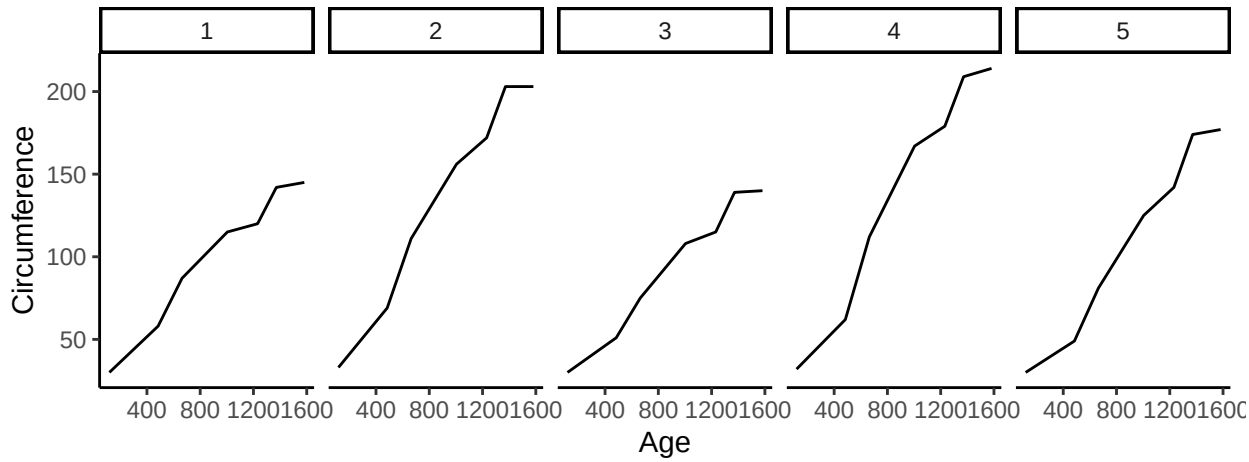


Figure 7.12: Trellis plot of trunk circumference (mm) for each tree against age (in days since December 1st, 1968).

The first few lines of data frame `orange` are:

```
  Tree    x circ  Season
1    1  118   30  Spring
2    1  484   58  Spring
3    1  664   87  Autumn
4    1 1004  115  Autumn
5    1 1231  120  Spring
6    1 1372  142  Autumn
```

where `Tree` is a factor with five levels, `x` is tree age (in days since 31 December 1968), `circ` is the trunk circumference (in mm), 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 converted to day of year, measurements were taken in either April/May (`Spring`) or September/October (`Autumn`).

Initially, we restrict the data set to tree 1 to demonstrate fitting cubic splines in ASReml-R. The model includes the intercept and linear regression of trunk circumference on x and an additional random term `spl(x)`. This random term includes a special design matrix with $7 - 2 = 5$ columns which relate to the vector, $\boldsymbol{\delta}$ whose elements $\delta_i, i = 2, \ldots, 6$ are the second differentials of the cubic spline at the knot points. The second differentials of a natural cubic spline are zero at the first and last knot points (Green and Silverman (1994)). The following code is used to fit this model:

```
orange.asr <- asreml(
  fixed = circ ~ x,
  random = ~spl(x),
  residual = ~units,
  knot.points = list(x = c(118, 484, 664, 1004, 1231, 1372, 1582)),
  subset = Tree == 1,
  data = orange
)
```

```
orange.asr$trace
```

```
                  1            2            3
LogLik  -20.9042520 -20.90127049 -20.90127049
Sigma2   48.4698131  49.15204735  49.15204735
DF        5.0000000   5.00000000   5.00000000
stepsz    0.5621388   1.00000000   1.00000000
spl(x)    0.1000000   0.09102007   0.07535652
units!R   1.0000000   1.00000000   1.00000000
```

In this example the spline knot points are specifically given in the `knot.points` argument. These extra points have no effect in this model fit as they are exactly the seven ages present in the data file. Hence, in this instance the analysis would be the same if the `knot.points` argument was omitted.

The estimated variance components and Wald statistics are:

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

```
        component std.error   z.ratio bound  %ch
spl(x)   3.703927  10.93709 0.3386573     P 20.8
units!R 49.152047  36.83725 1.3343028     P  0.0
```

```
ww <- wald(orange.asr, denDF = "numeric", ssType = "conditional")
```

```
            Df denDF  F.inc  F.con Margin           Pr
(Intercept)  1   3.5 1377.0  17.96        0.0174544178
x            1   3.5  216.7 216.70      A 0.0002781746
```

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

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

And the first and last few rows are:

```
    x predicted.value std.error    status
1 118        30.75934  6.043879 Estimable
2 150        33.72253  5.701909 Estimable
3 200        38.35142  5.225972 Estimable
4 250        42.97744  4.833550 Estimable
5 300        47.59886  4.529993 Estimable
6 350        52.21392  4.308913 Estimable
```

```
      x predicted.value std.error    status
30 1350        134.3050  3.707470 Estimable
31 1372        135.8261  3.782961 Estimable
32 1400        137.7490  3.899193 Estimable
33 1450        141.1511  4.172183 Estimable
34 1500        144.5228  4.546196 Estimable
35 1582        150.0168  5.385572 Estimable
```

The `design.points` argument used above adds the nominated points to the design matrix for prediction purposes (Figure 7.13). Note that `design.points` could have been included in the call to `asreml.options()` instead of in `predict()`. If omitted from either `predict()` or `asreml.options()` a default set of points for prediction purposes would have been generated. The

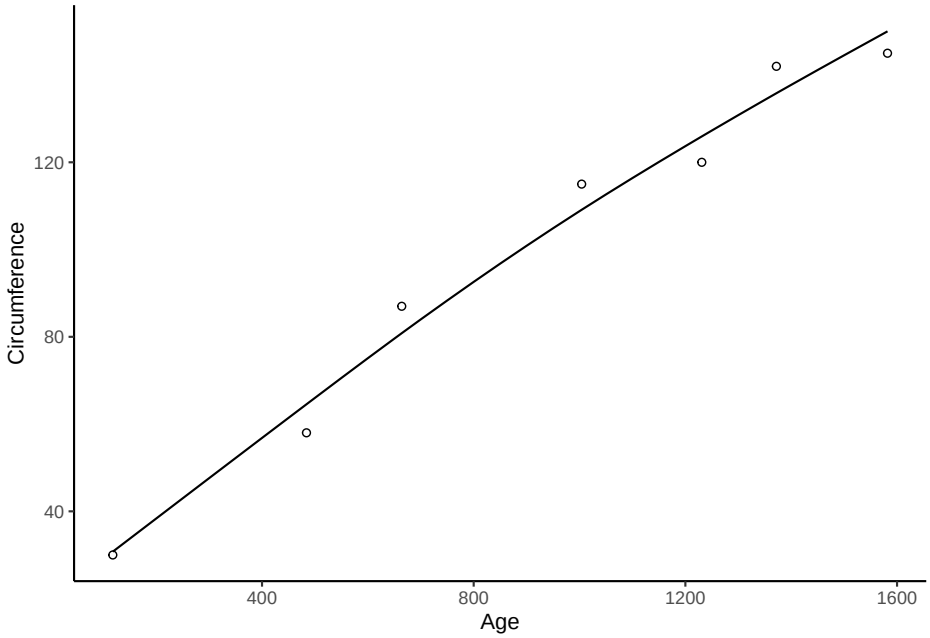


Figure 7.13: Fitted cubic smoothing spline for tree 1

Table 7.11: ANOVA decomposition for the orange data

stratum	decomposition	type	df or ne
(Intercept)	1	f	1
`Age`			
	`x`	f	1
	`spl(x)`	r	5
	`residual`	r	7
`Tree`			
	`Tree`	rc	5
`Age:Tree`			
	`x:Tree`	rc	5
	`spl(x):Tree`	r	25
`error`		r	

REML estimate of the smoothing constant and the fitted cubic smoothing spline (Figure 7.13) indicate there is some nonlinearity. The four points below the line are the spring measurements.

An analysis of variance decomposition for the full data set is given in Table 7.11, following Verbyla et al. (1999). 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, which is consistent with Verbyla et al. (1999). In this sense, the tree deviation splines play a role in modelling the conditional

curves for each tree and variance modelling. The intercept and slope for each tree are included as random coefficients (denoted by **rc** in Table 7.11). Thus, if $\boldsymbol{U}^{5\times 2}$ is the matrix of intercepts (column 1) and slopes (column 2) for each tree, then we assume that

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

where $\boldsymbol{\Sigma}$ is a 2×2 symmetric positive definite matrix. Non-smooth variation can be modelled at the overall mean (across trees) level and this is achieved by including the factor `dev(x)` as a random term. This full model is:

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

Note that the command `str()` will apply the direct product variance defined by $\boldsymbol{\Sigma}_2 \otimes \boldsymbol{I}_5$ to the combined set of terms `Tree` and `x:Tree`. In this case Σ_2 is a diagonal matrix of dimension 2 but other structures can be defined.

Table 7.12 presents the sequence of fitted models. We stress the importance of model building in these settings, where we generally start with relatively simple variance models and update to more complex variance models if appropriate. Note that the REML log-likelihood values 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 $\boldsymbol{\Sigma}$ for the intercept and slope for each tree as a diagonal matrix as there is no point including a covariance component between the intercept and slope if the variance component(s) for one (or both) is zero. Model 1 also does not include a non-smooth component at the overall level (that is, `dev(x)`).

The ASReml-R call and variance components for model 1 are:

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

Table 7.12: Sequence of models fitted to the `orange` data

	model					
term	1	2	3	4	5	6
`Tree`	y	y	y	y	y	y
`x:Tree`	y	y	y	y	y	y
`cov(Tree, x:Tree)`	n	n	n	n	n	y
`spl(x)`	y	y	y	y	n	y
`spl(x):Tree`	y	y	y	n	y	y
`dev(x)`	n	y	y	n	n	n
`Season`	n	n	y	y	y	y
REML log-likelihood	-97.78	-94.07	-87.95	-91.22	-90.18	-87.43

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

```
                          component    std.error  z.ratio bound %ch
spl(x)                 6.398248e+02 4.130439e+02 1.549048     P 0.3
Tree+x:Tree!diag(2)_1 3.048865e+01 2.457266e+01 1.240755     P 0.4
Tree+x:Tree!diag(2)_2 5.980326e-04 4.240080e-04 1.410428     P 0.3
spl(x):Tree            7.101353e+00 4.915670e+00 1.444636     P 0.7
units!R                6.374557e+00 3.658904e+00 1.742204     P 0.0
```

The fitted curves from this model are shown in Figure 7.14. The fit is unacceptable because the spline has picked up too much curvature, suggesting there may be systematic non-smooth variation at the overall level. This can be formally examined by including the `dev(x)` term as a random effect.

```
orange6.asr <- asreml(
  fixed = circ ~ x + Season,
  random = ~str(~Tree + x:Tree, ~us(2, init = c(5.0, -0.01, 0.0001)):id(5)) +
            spl(x) + spl(x):id(Tree),
  residual = ~units,
  knot.points = list(x = c(118, 484, 664, 1004, 1231, 1372, 1582)),
  data = orange
)
```

Model 2 increased the REML log-likelihood by 3.70 ($P < 0.05$) with the `spl(x)` smoothing constant approaching the boundary. The `Season` factor provides a possible explanation. When included in Model 3, `Season` has a Wald statistic of 107.3 ($P < 0.01$) and `dev(x)` becomes bounded. The spring measurements are lower than the autumn measurements so growth is slower in winter. Models 4 and 5 successively examined each term, indicating that both smoothing constants are significant. Model

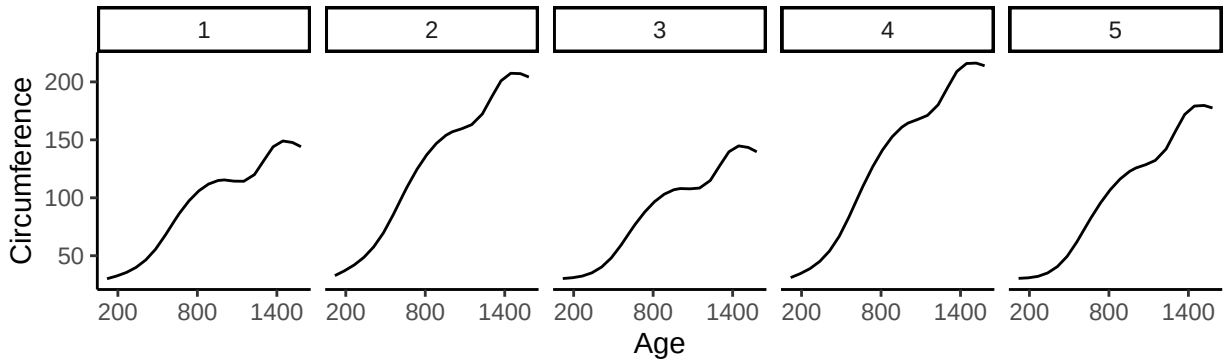


Figure 7.14: Plot of fitted cubic smoothing spline for model 1

6 includes the covariance parameter between the intercept and slope for each tree; this ensures that the model will be translation invariant. This model requires care in the choice of starting values. The ASReml-R call, illustrating an alternative method for specifying initial values, and the estimated variance components for model 6 are:

```
                        component    std.error  z.ratio bound %ch
spl(x)                6.398248e+02 4.130439e+02 1.549048     P 0.3
Tree+x:Tree!diag(2)_1 3.048865e+01 2.457266e+01 1.240755     P 0.4
Tree+x:Tree!diag(2)_2 5.980326e-04 4.240080e-04 1.410428     P 0.3
spl(x):Tree           7.101353e+00 4.915670e+00 1.444636     P 0.7
units!R               6.374557e+00 3.658904e+00 1.742204     P 0.0
```

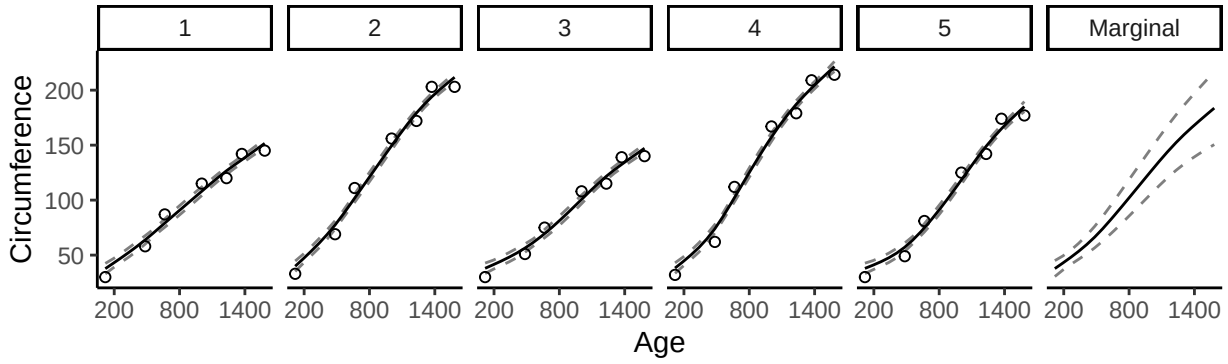


Figure 7.15: Fitted values adjusted for 'Season'

The fitted values for individual trees (adjusted for `Season`) from Model 6 together with a marginal prediction and approximate confidence intervals (2× standard error of prediction) are shown in Figure 7.15. The conclusions from this analysis are quite different from those obtained by the nonlinear mixed effects analysis. The individual curves for each tree are not convincingly modelled by a logistic function. There is a distinct pattern in the residuals shown in Pinheiro and Bates (2000, 340), which is consistent for all trees; this is modelled here by the `Season` term.

Appendix A

Some technical details about model fitting in ASReml-R

A.1 Sparse *versus* dense

The terms in the linear mixed model are partitioned into two sets; a dense set and a sparse set. The partition is defined by the fixed formula; all terms in the `fixed` formula are included in the dense set, while all 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.

As indicated, random effects are, by default, part of the sparse set. However, genetic relationship matrices, for example, are typically quite large (in the order of several thousand) and dense, and it can be more efficient to process such terms as dense. The `dense` formula component of the `asreml.options()` list can be used to include terms in the `random` formula in the dense set of equations for processing. The `dense` formula can also be given directly as an argument to `asreml()`. In the following example, we present two ways to include the equations for `clonefv` in the dense set for processing.

```
asreml.options(dense = ~vm(clonefv, nassau.grm))
nassau.asr <- asreml(
  fixed = ht6 ~ CultureID + CultureID:Rep,
  random = ~vm(clonefv, nassau.grm) + ide(clonefv) + Rep:IncBlock,
  residual = ~units,
  data = nassau
)
```

```
nassau.asr <- asreml(
  fixed = ht6 ~ CultureID + CultureID:Rep,
  random = ~vm(clonefv, nassau.grm) + ide(clonefv) + Rep:IncBlock,
  dense = ~vm(clonefv, nassau.grm),
  residual = ~units,
  data = nassau
)
```

A.2 Ordering of terms in ASReml-R

Solutions for the fixed and random effects in linear mixed model analysis using ASReml-R are obtained by solving the corresponding mixed model equations in the numerical routines (Gilmour et al. 1995). The sparse equations are processed first after being re-ordered to retain sparsity during solution. If `keep.order = FALSE`, the remaining equations are processed with main effects before interactions and low-order interactions before higher ones so that normal marginality of terms is achieved. The order of effects in the solution vector(s) in the returned object reflects the order of processing.

A.3 Aliasing and singularities

A singularity occurs when there is either:

- a linear dependence in the model and therefore no information left to estimate the corresponding effect,
- no data for a given fixed effect, or
- 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.

Important 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, under REML, 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-parameterized models. It is assumed that `Var` is defined with four levels, `Trt` with three levels and `Rep` with three levels, and that all `Var:Trt` combinations are present in the data.

Table A.1: Examples of aliasing

model	number of singularities	description
`fixed = y ~ -1 + Var, random = ~ Rep`	0	`Var` fully fitted
`fixed = y ~ Var, random = ~ Rep`	1	first level of `Var` dropped
`fixed = y ~ -1 + Var + Trt, random = ~ Rep`	1	`Var` fully specified, first level of `Trt` dropped from the models
`fixed = y ~ Var + Trt + Var:Trt, random = ~ Rep`	8	first level of both `Var` and `Trt` dropped from the model, together with subsequent interactions
`fixed = ~ Var + Trt, random = ~ Rep, sparse = ~ Var:Trt`	8	`Var:Trt` fully specified; `(Intercept)`, `Var` and `Trt` completely singular and dropped from the model

Appendix B

Description of features in ASReml-R

In this section we describe all arguments, options and output lists associated with ASReml-R. A few of these are illustrated in more detail.

B.1 Model formula terms

- `and(obj, t)` Add `t` times the design matrix for `obj` to the previous columns.
- `at(obj, l)` Form conditioning covariables for the levels $l = 1, \ldots, k$ of factor `obj`. That is, it defines a binary variable which is `1` if the factor `obj` has level `l` for the observation. For example, to fit a row factor only for site 3, use the expression `at(Site, 3):row`. Note that if `l` is numeric, then the level of `f` is chosen as the `l`th in factor (sorted) order. Note also that when used with spline terms, such as `at(f, 2):spl(x)` then the knot points are derived from **all** knots of factor f, not just level 2.
- `C(obj, contr)` Define contrasts among the levels of `obj` from the coefficients in vector `contr`.

  For example, the following vector `L` is a contrast based on the factor `Nitrogen` that has four levels:

```
L <- c(3, 1, -1, -3)
oats.asr <- asreml(
  fixed = yield ~ C(Nitrogen, L),
  data = oats
)
```

  This contrast is defined by the term `C(Nitrogen, L)` in the model call. The length of the contrast vector (pre-defined as `L` in this case, although `c(3, 1, -1, -3)` could replace `L` in the call) must be equal to the number of levels in the factor. Missing values in the factor are excluded in forming the contrast.
- `con(obj)` Apply sum-to-zero constraints to factor `obj`.

- `dev(x)` Fit variate `x` as a factor; hence, it forms a factor with a level for each unique value of the covariate {x}; typically used for spline deviations.

- `dsum( term | obj)` Form direct sum operation of `term` for the levels of `obj`. Used in `residual` to define multiple *sections*.

- `gpf(obj)` Include the term defined in the `combine` argument in the model. It forms a new factor named `obj` from an existing factor by merging a subset of its levels. The name `obj` must also appear as a component of the `combine` argument to `asreml()` where the existing factor and the levels to merge are defined with a call to `levels()`. The function `Levels` is defined as: `function(f, x)` where `f` is the name of an existing factor and `x` is a vector of length `length(levels(f))` defining the levels of `f` to merge. For example, if `Site` has levels `"1"`, `"2"` and `"3"`, `combine = list(A = Levels(Site, c("1", "2", "1")))` creates a new factor `A` with levels `"1"` and `"2"` by merging levels `"1"` and `"3"` of `Site`, and would be included in the model as `gpf(A)`. While the actions of `gpf()` can be duplicated outside `asreml()`, `gpf()` is necessary if `predict()` is to be used.

- `grp(obj)` Include the term defined by `obj` in the `group` argument in the model. This function groups contiguous columns of the data frame `data` to be treated as a single factor named `obj`. The columns of `data` are identified by a character or numeric vector component `obj` of the `group` argument to `asreml()`.

- `leg(x, t)` Legendre polynomials to degree `t`. It forms `t` Legendre polynomials from the values in `x`; the intercept is excluded if `t` is negative. For example, `leg(time, 2)` is a factor with three columns: a constant in the first, centred and scaled linear covariate in the second and centred and scaled quadratic covariate in the third. `leg()` could be interacted with a design factor to fit random regression models.

- `lin(obj)` Fit factor `obj` as a variate. The function is defined for `obj` being a simple factor, trait and units. The `lin(obj)` function does not centre or scale the variable.

- `ma(obj)` Form a moving average (order 1) design matrix from factor `obj`.

- `mbf(obj)` Include the covariates defined by `obj` in the `obj` argument as a factor. It includes this set of covariates to be fitted as a single term in a similar way to `grp`. The name `obj` must also appear as a component of the `mbf` argument to `asreml()` where the data frame holding the covariates is identified along with a key field for merging records with those in `data`.

- `pol(x, t)` Orthogonal polynomials to degree `t` constructed from the values in `x`; the intercept is excluded if `t` is negative. For example, `pol(time, 2)` is a factor with three columns: a constant in the first, centred and scaled linear covariate in the second and centred and scaled quadratic covariate in the third. `pol()` could be interacted with a design factor to fit random regression models.

- `pow(obj, p, offset)` Create the term: $(\mathbf{offset} + \mathbf{obj})^{\mathbf{p}}$.

- `sbs(obj)` Include the term defined in the `prune` argument in the model. This function forms a new factor named `obj` from an existing factor by selecting a subset of its levels. The name `obj` must also appear as a component of the `prune` argument to `asreml()`, where the

existing factor and the subset of levels to select are defined with a call to `Subset()`. The function `Subset` is defined as: `function(f, x)` where `f` is the name of an existing factor and `x` is a character or numeric vector of levels to select. For example, `prune = list(A = Subset(Site, c(2, 3)))` creates a new factor `A` by selecting the second and third levels of `Site`, and would be included in the model as `sbs(A)`. While the actions of `sbs()` can be duplicated outside `asreml()`, `sbs()` is necessary if the method `predict()` is to be used.

- `spl(x, k)` The *random* component of a cubic spline based on covariate `x`. `spl(x)`, `dev(x)` and possibly `lin(x)` are used when fitting cubic splines. The cubic spline is composed of a random nonlinear component imposed on a linear trend. It is fitted by including a special random factor, `spl(x)`, and the fixed covariate (`x`) in the linear model. Knot points (`k`) are placed at the design points if `length(unique(x))` $<$ `k` otherwise there are `k` equally-spaced knot points over the range of `x`. The default for `k` is 50. If `k` is omitted then knots can be set in `asreml.options()`. Also, explicit knot points are set in the `knot.points` argument to `asreml()`.
- `uni(obj)` Include the term defined in the `uid` argument in the model.

B.2 Reserved names/words

The following table presents a list of reserved names from ASReml-R that are used as part of the model formula for fixed and/or random effects.

Table B.1: Summary of reserved names

Term	Purpose	Usage
`mv`	It fits missing values as covariates. An example of its use is in spatial analyses, for example, where computing advantages arising from a balanced spatial layout can be exploited. Missing values in the response are handled in two ways using the `na.method()` function. If `na.action = na.method(y = "omit")`, records containing missing values in the response are deleted. If `na.action = na.method(y = "include")`, missing values are estimated and a factor labelled `mv` is included in the model frame. If a variate labelled `mv` already exists in the data frame it will be overwritten. For a multivariate analysis, missing values must currently be included.	`f`
`trait`	Used with multivariate data to fit the individual trait means. It is interacted with other factors to estimate their effects for all traits. It is formally equivalent to the intercept (`1`), but it is a more natural label for use with multivariate data. If a variate labelled `trait` already exists in the data frame it will be overwritten.	`f`, `r`
`units`	A factor with a level for each experimental unit; allows a second error term to be explicitly fitted.	`r`

B.3 The `asreml()` arguments

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

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

  `function(f, x)`

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

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

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

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

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

- `family` A list of functions and expressions for defining the link and variance functions.

  Supported families are: *gaussian*, *inverse Gaussian*, *binomial*, *negative binomial*, *poisson*, *Gamma* and *multinomial*. Family objects are generated from the `asreml` family functions which prefix the usual function names with "`asr_`"; for example `asr_gaussian()`, `asr_binomial()`, etc. In addition to the link argument, these functions take an additional `dispersion` argument and a `total` argument where relevant; for example:

  `asr_binomial(dispersion = 1.0, total = counts)`.

  The default for `asr_gaussian()` is `dispersion = NA`, which implies that `asreml` will estimate the dispersion parameter, otherwise the scale is fixed at the nominated value.

- `fixed` A formula object specifying the fixed terms in the model, with the response on the left of a $\sim$ operator, and the terms, separated by + operators, on the right. If `data` is given, all names used in all formulae should appear in the data frame. A model with the intercept as

the only fixed effect can be specified as ~ 1; there must be at least one fixed effect specified. If the response (y) evaluates to a matrix then a factor `trait` with levels `dimnames(y)[[2]]` is added to the model frame, and must be explicitly included in the model formulae (default = `1`).

- `G.param` Either:
  - A list object derived from the `random` formula, holding initial parameter estimates and boundary constraints for each term, or
  - A character string naming a comma-delimited file with a header line and three columns for the variance component name, initial value and constraint code, respectively. This file can be created using the `start.values = TRUE` argument; the internal list object is then generated from the contents of this file.

  On termination, `G.param` is updated with the final `random` component estimates.
- `group` A named list where each component is a numeric vector specifying contiguous fields in `data` that are to be considered as a single term. The component names can then appear in the `asreml` model formulae using the `grp()` special function. The default is an empty list.
- `knot.points` A named list where each component is a vector of user-supplied knot points for a particular spline term; the component name is the object of the `spl()` model function.
- `last` A named list restricting the order equations are solved in the sparse partition for the nominated model terms. Each component of the list is named by a model term and contains a scalar `n` specifying that the first `n` levels of the term will be solved after all others in the sparse set. It is intended for use when there are multiple fixed terms in the sparse equations so that `asreml` will be consistent in which effects are identified as singular. A maximum of three factor/level pairs can be specified.
- `mbf` A named list specifying sets of covariates to be included with one or more `mbf()` model functions. Each component of the list must in turn contain components named `key` and `cov`, where `cov` is a character string naming the data frame holding the covariates, and `key` is a character vector of length two naming the columns in `data` and `cov`, respectively, used to match corresponding records in the two data frames. The default is an empty list.

  The covariate data frame required for `mbf` can be supplied using the component `cov`, as shown in the following example:

```
naf.asr <- asreml(
  fixed = yield ~ type,
  random = ~mbf(A) + Variety,
  residual = ~ar1v(column):ar1(row),
  mbf = list(A = list(key = c("Variety", "V1"), cov = "naf31H")),
data = naf
)
```

  fits a function associated with factor `A` defined in data frame `naf31H`.

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

  `mef = list(MM = snp.mat)`

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

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

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

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

  The motivation is to reduce the size of the `asreml` object in large examples. The data as used in the fit is needed for some post fitting methods like `resid` and `plot`, but automatically including it in the object adds to the size of the object and the `.RData` file, particularly if the data alone is hundreds or thousands of megabytes. The `RDS` file is a convenient way to keep the data (model) frame and all its properties outside the `.RData` workspace. This is illustrated below:

```
oats.asr <- asreml(
  fixed = yield ~ Variety * Nitrogen,
  data = oats,
  model.frame = "oats.RDS"
)
```

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

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

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

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

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

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

  `function(f, x)`

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

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

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

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

```
grass.asr <- asreml(
  fixed = cbind(y1, y3, y5, y7, y10) ~ trait + Tmt + trait:Tmt,
  residual = ~ id(units):exp(trait),
  pwr.points = list(trait = c(1, 3, 5, 7, 10)),
  data = grass
)
```

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

- `R.param` Either:

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

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

- `random` A formula object specifying the random effects in the model. This argument has the same general characteristics as `fixed`, but there can be no left side to the $\sim$ operator. Variance structures imposed on random terms are specified using special model functions (default = `NULL`).

- `residual` A formula object specifying the residual model; any term specified on the left of the $\sim$ expression is ignored. The default is $\sim$ `units`, where the reserved word `units` is defined as `seq(1,nrow(data))` and is automatically included in the model frame. Variance models for

the residual component of the model can be specified using special model functions (default = `NULL`).

For single-section univariate models, the residual variance model determines the computational mode: If the residual variance model specifies a correlation structure (includes `id()`), then the model is fitted on the gamma scale, otherwise the model is fitted on the sigma scale. The default is `id(units)` if not explicitly specified.

- `sparse` A formula object, specifying the fixed effects for which the full variance-covariance matrix is not required. This argument has the same general characteristics as `fixed`, but there can be no left side to the $\sim$ expression. Wald statistics are not available for sparse fixed terms in order to reduce the computing load (default = `NULL`).

- `start.values` If `TRUE`, `asreml` exits prior to the fitting process and returns a list of length three: the `G.param` and `R.param` lists, and a data frame (containing variance parameter names, initial values and boundary constraints). Initial values or constraints can then be set in the list or data frame objects (default = `FALSE`).

  If this is a character string, then a file of that name is created and the data frame object containing initial parameter values is written out in comma-separated form. This file can be edited externally and subsequently specified in the `G.param` or `R.param` arguments.

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

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

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

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

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

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

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

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

  For example, the following `vcc` matrix:

1 1
2 1
2 2
3 1

is equivalent to the `vcm` matrix:

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

- `vcm` A matrix defining relationships among variance parameters. The matrix has a row for each original variance parameter and a column for each new parameter. The default is the identity matrix, that is, no action. See function `vcm.lm` for further information and an example.
- `wald` A named list with three components: `denDF`, `ssType` and `Ftest`.
  - `denDF`: A character string from the options: `"none"`, `"numeric"`, `"algebraic"`, and `"default"` specifying the calculation of approximate denominator degrees of freedom. The option `"none"` is to suppress the computations. Algebraic computations are not feasible in large analyses; use `"default"` to automatically choose numeric or algebraic computations depending on problem size. The denominator degrees of freedom are calculated according to Kenward and Roger (1997) for terms in the `fixed` model formula (default = `"none"`).
  - `ssType`: It can be `"incremental"` for incremental sum of squares or `"conditional"` for F-tests that respect both structural and intrinsic marginality (default = `"incremental"`).
  - `Ftest`: A one-sided formula of the form $\sim$ `test_term | background_terms` specifying a conditional Wald test of the contribution of `test_term` conditional on those fixed terms listed in `background_terms`, and those in the `random` and `sparse` model formulae.
- `weights` A character string or name identifying the column of `data` to use as weights in the fit.
- `...` Additional arguments to `asreml()` directed at `asreml.options()`, such as: `maxit`, `workspace`, `pworkspace`, `fixgammas`, `trace`, `aom`.

B.4 The `asreml` object list

The library `asreml` once run provides an object of class `asreml` with a rich list of elements that can be used to extract output or perform additional processing based on the fitted linear model. This section provides additional details of the elements of this list.

- `ai` The inverse average information matrix of the variance parameters. A `Matrix` class object, sub-class `dspMatrix` is returned.
- `call` An image of the `asreml` function call.

- `Cfixed` Reflexive generalized inverse of the coefficient matrix of the mixed model equations relating to the dense fixed effects (if , a matrix of class `Matrix`, sub-class `dspMatrix` is returned).

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

- `coefficients` A list with three components named `fixed`, `random` and `sparse` containing the solutions to the mixed model equations corresponding to the E-BLUEs of the fixed effects, the E-BLUPs of the random effects, and the solutions corresponding to the sparse fixed effects, respectively. The coefficients are labelled by a concatenation of factor name and level separated by `"_"`.

- `Csparse` If is not `NULL`, the non-zero elements of the reflexive generalised inverse matrix of the coefficient matrix for the sparse stored model terms nominated in the `Csparse` formula. A matrix in triplet form giving the row, column and non-zero elements.

- `design` The design matrix as a sparse `Matrix` of class `dgCMatrix` if `asreml.options(design = TRUE)`.

- `deviance` The deviance from the fit.

- `factor.names` A character vector of term names appearing in the model.

- `fitted.values` A vector containing the fitted values from the model, obtained by transforming the linear predictors by the inverse of the link function.

- `formulae` A list object containing the `fixed`, `random`, `sparse` and `residual` formula arguments to `asreml`.

- `G.param` A list object containing the constraints and final estimates of the variance parameters relating to the random part of the model. This object may be used as the value of the `G.param` argument to provide initial parameter estimates to `asreml`.

- `hat` The diagonal elements of the matrix $WC^{-1}W^T$, the *extended* hat matrix. This is the linear mixed effects model analogue of $X(X^TX)^{-1}X^T$ for ordinary linear models.

- `license` A character string containing the license information. The string has embedded new-line characters and is best formatted through `cat()`.

- `linear.predictors` The linear fit on the link scale.

- `loglik` The log-likelihood at completion of the `asreml` call.

- `meff` Regressor scores (marker effects) if nominated in the `mef` list argument.

- `mf` The model frame with the data as a `data.table` object with numerous attributes from the model specification. Inspect for details.

- `nedf` The residual degrees of freedom, `length(y) - rank(X)`.

- `noeff` A vector containing the number of effects for each term.
- `nwv` The number of working variables.
- `predictions` If `predict` is not `NULL`, a list object with components `pvals, sed, vcov` and `avsed`. The `predictions` component only is returned by the `predict` method for `asreml` objects.
- `R.param` A list object containing the constraints and final estimates of the variance parameters relating to the error structure of the model. This object may be used as the value of the `R.param` argument to provide initial parameter estimates to `asreml`.
- `residuals` A single column matrix containing the residuals from the model.
- `score` The score vector of length number of random parameters.
- `sigma2` The REML estimate of the scale parameter.
- `trace` A numeric matrix recording the convergence sequence for each random component, as well as the log-likelihood, residual variance and residual degrees of freedom.
- `vcoeff` A list with three components named `fixed`, `random` and `sparse` containing the unscaled variances of the coefficients. The actual variances are calculated as `vcoeff*object$sigma2` and returned by the `summary` function.
- `vparameters` The vector of variance components estimates from the model fit.
- `vparameters.con` A numeric vector identifying the boundary constraint applied to each variance parameter at termination. Common values are 1, 3 and 4 for **P**ositive, **U**nconstrained and **F**ixed, respectively.
- `vparameters.pc` Percentage change in `gammas` parameters over the last iteration.
- `vparameters.type` A numeric vector identifying the variance parameter types. Numeric values are used internally and the character codes as used by the `own()` variance model can be obtained from the function `vpt.char`.
- `yssqu` A vector of incremental sums of squares for (dense) fixed terms.

B.5 The `asreml.options()` arguments

In `asreml.options()` the less frequently used settings are set per session outside the `asreml()` function call in an `options environment` by a call to `asreml.options()`. With no arguments, `asreml.options()` returns a list of settings that can be altered; arguments are a sequence of `name = value` pairs. The full list of options follows and can be displayed in R by typing `help(asreml.options)`.

- `about` If `TRUE`, the date packaged is printed for identification (default = `FALSE`).
- `ai.loadings` Controls modification to Average Information (AI) updates of loadings in extended factor-analytic (`fa(, k)`) models. After `asreml` calculates updates for variance parameters, it checks whether the updates are reasonable and sometimes reduces them over and above any `step.size` shrinkage. The extra shrinkage has two levels: loadings that change sign are restricted to doubling in magnitude; and if the average change in magnitude of loadings is greater than 10-fold, they are all shrunk. Unless the user specifies constraints, `asreml` sets them and rotates the loadings each iteration.

  When `ai.loadings =` i is specified ($i = -1$ is no action), it also prevents AI updates of some loadings during the first i iterations. For $f > 1$ factors, only the last factor is estimated (conditional on the earlier ones) in the first $f - 1$ iterations. Then pairs, including the last, are estimated until iteration i (default = `0`).
- `ai.penalty` This refers to the algorithm for updating loadings in factor analytic (`fa`) models. The present strategy modifies the average information matrix by increasing the diagonal elements pertaining to loadings by a percentage, p. The default is to start with $p = 10\%$ and reduce it by 1 or 2% each iteration down to 1%. If the starting values are poor, 10% may not be a sufficient initial retardation. If it appears the updates are unreasonable, the value of p is increased by 10%. After the penalty has reduced to 1%, it is further reduced to 0.2%. `ai.penalty` can be set to 0 if desired (default = `10`, corresponding to 10%).
- `ai.sing` If `TRUE`, forces continuation if a singularity is detected in the average information matrix. Variance components are reported to help identify singularity but these are often incorrect (default = `FALSE`).
- `aodev` If `TRUE`, return an analysis of deviance (default = `FALSE`).
- `aom` If `TRUE`, return standardized conditional residuals and standardized conditional BLUPs in the `aom` component of the `asreml` object (default = `FALSE`).
- `Cfixed` If `TRUE`, return the computed part of the C^{-1} matrix in component `Cfixed`. The inverse coefficient matrix is fully formed for terms in the dense set (default = `FALSE`).
- `colourise` If `TRUE`, the header text produced by methods such as `wald` and `predict` will be displayed in a different colour if supported by the output terminal device (default = `TRUE`).
- `Csparse` If a formula is specified, return the computed part of C^{-1} for those terms given in the formula. The library `asreml` does not compute the whole of C^{-1}, only that which is sufficient to calculate the REML solution (default = `Csparse = NULL`).
- `debug` Return internal data structures for helping debugging (default = `FALSE`).
- `dense` Include the equation(s) for the term(s) in the formula in the dense set. This results in faster processing if the term is associated with a known dense inverse relationship matrix (default = `dense = NULL`).
- `design` If `TRUE`, return the design matrix for component `design` of the `asreml` object. This option might be used, for example, in generating design matrices that can then be used in post-processing (default = `FALSE`).

- `drop.unused.levels` If `TRUE`, levels of simple factors that do not appear in the data are dropped (default = `TRUE`).
- `eqorder` Set the algorithm used for ordering the mixed-model equations prior to solution. The argument `eqorder = -1` processes the equations in *user* order; generally this will run much slower, if at all, in real time for large analyses (default = `3`).
- `extra` Forces other `mod(extra, maxit)` iterations after apparent convergence (default = `0`).
- `fail` If `"hard"`, fatal errors will terminate execution, otherwise if `"soft"` such conditions will be reported as warnings, allowing testing runs, for example, to continue. In both cases the `converge` component of the `asreml` object will be set to `FALSE` and the results will be erroneous (default = `"hard"`).
- `fixgammas` If `TRUE`, all variance parameters are constrained to be fixed at their starting values (default = `FALSE`).
- `font.scale` Scale axis text and labels (relative to the `asreml` default settings) in the graphs generated by `plot()` (default = `1.0`).
- `gammaPar` If `TRUE`, single-section models will be fitted using the `gamma` parameterization irrespective of whether the `residual` formula specifies a correlation or variance model. The default behaviour for single-section models is to fit on the `gamma` scale if the `residual` formula specifies a correlation structure, and on the `sigma` scale if the `residual` formula specifies a variance structure (default = `FALSE`).

  This option allows the user to change between the sigma and gamma parameterizations for model fitting. There are sometimes advantages in using a gamma parameterization in terms of providing appropriate initial values. If the residual model has only one variance parameter then setting `gammaPar = TRUE` within `asreml.options()` switches from the sigma parameterization to the gamma parameterization in fitting the mixed model.
- `glmminloop` Set the number of inner iterations performed in an iteratively weighted least squares analysis. These estimate the effects in the linear model for the current set of variance parameters; outer iterations are the average information updates to the variance parameters. The default is to perform 4 inner iterations in the first round and 2 in subsequent rounds of the outer iteration. Set to 2 or more to increase the number of inner iterations (default = `1`).
- `grid` A logical vector of length 1 or `length(design.points)` (see `predict`) controlling the expansion of coordinates for two-dimensional kriging. For a given term, the coordinates for prediction in two dimensions (x, y) are given as a list of two vectors or a two-column matrix component of `design.points`. If `TRUE`, the coordinates are expanded to form an (x, y) grid of all possible combinations, otherwise the columns of the matrix are taken in parallel (default = `TRUE`).
- `keep.order` If `TRUE`, the order of terms in the `fixed` formula is retained. Set to `TRUE` if the special model function `and()` is present (default = `FALSE`).
- `knots` The number of knot points for spline terms to consider. For a variate `x`, the number of knot points is `min(length(unique(x)), knots)` (default = `50`).

- `license_id` If `0`, a new license_id is created for identification (default = `0`).
- `maxit` Maximum number of iterations to stop fitting (default = `13`).
- `nsppoints` Value that influences the number of points used when predicting splines and polynomials. The design matrix generated by the `pol(x)` and `spl(x)` functions are modified to include extra rows for points used in prediction. The range of `x` is divided by `nsppoints - 1` to give a step size i. For each point p in x, a predict point is inserted at $p + i$ if there is no data value in the interval $[p, p + 1.1i]$.

  The argument `nsppoints` is ignored if the `predict()` argument `design.points` is set (or the `design.points` component of the `predict` list argument to `asreml()` is not empty). This process also affects the number of levels identified by `dev(x)` (default = `21`).
- `oscillate` If `TRUE`, the test for oscillating log-likelihood is implemented (default = `TRUE`).
- `pworkspace` Sets the workspace needed by the `predict()` method; follows the same convention as `workspace`. Ignored if the `predict` argument to `asreml()` is not set. Note that the total workspace used for prediction is: `workspace + pworkspace` (default = `"128mb"`).
- `pxem` Selection of the (PX)EM update strategy for unstructured (`us`) variance models when average information updates cause them to be non-positive definite (see `uspd`) (default = `1`). Valid options are:

  1 Standard EM + 10 local EM steps
  2 Standard EM + 10 local PXEM steps
  3 Standard EM + 10 local EM steps*
  4 Standard EM + 10 local PXEM steps*
  5 Standard EM only
  6 Single local PXEM
  7 Standard EM + 1 local EM step
  8 Standard EM + 1 local PXEM step

  *Options 3 and 4 cause all `us` structures to be updated by (PX)EM if any particular one requires EM updates.
- `random.order` If option `"noeff"` the terms in the `random` and `sparse` formulae are reordered in increasing number of effects. This is almost always desirable, especially if the stratum variance decomposition is required. Other options are `"user"` to retain the order given in the model specification, or `"R"` for the default `R` rules (default = `"noeff"`).
- `rotate.fa` If `FALSE`, `asreml()` initially constrains the first $k - 1$ loadings for higher order ($k > 1$) factors in factor analytic models to zero. If constraints are not set for factor analytic models with more than one factor, `asreml()` will set them internally and rotate the loadings each iteration (`rotate.fa = TRUE`). This option also modifies the action of `update.Gcon` such that rotation, if specified, is applied on an update (default = `FALSE`).

  Constraints are required in the $\mathbf{\Gamma}$ matrix for $k > 1$ for identifiability in `fa(, k)` models. These are automatically set unless the user ensures identifiability by constraining one parameter in the second column, two in the third column, and so on. With `rotate.fa = FALSE`, ASReml-R

fixes the $j = 1, ..., i-1$ loadings for the i-th factor ($2 \leq i \leq k$) to zero and their corresponding boundary constraints to `FALSE`. The total number of constraints is: $k(k-1)/2$.

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

- `scale` Overall scale parameter (default = `1.0`).
- `score` If `TRUE`, the score vector is returned in a component `score` of the `asreml` object (default = `FALSE`).
- `spline.scale` When forming a design matrix for a `spl()` term, a standardized scale is used. Setting `spline.scale = 1` forces `asreml` to use the scale of the variable. The value `-1` is recommended in most cases (default = `-1`).
- `spline.step` A list with components named `spl`, `dev` and `pol` specifying the resolution for spline deviations and polynomial functions, respectively. Points closer together than `1/spline.step` of the range will be treated as a single point (default `spline.step = list(spl = 10000, dev = 10000, pol = 10000)`).
- `step.size` Value for updating shrinkage factor. It reduces the update step sizes of the variance parameters. The step size is incremented each iteration to a maximum of 1.0 (default = `0.316`).
- `threads` The maximum number of threads to be used with OMP parallel processing. `asreml` will use all threads available up to the maximum number specified. The value `-1` will use all available threads (default = `-1`).
- `tol` A vector of length two that modifies the sensitivity of `asreml` to detect singularities in the mixed model equations. This is intended for the rare occasions when singularities are detected after the first iteration (default = `c(0, 0)`).

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

  If the problem of later singularities arises because of the low coefficient of variation of a covariable, it may be advisable to centre and rescale the covariable. If the degrees of freedom are correct in the first iteration, the problem lies with the variance parameters and a different variance model (or constraint) is needed.
- `trace` If `TRUE`, convergence monitoring of the current fit is reported in the console (default = `TRUE`).

- `update.Gcon` If `TRUE`, the constraint status of variance parameters in the `G.param` list component on termination is updated; this may influence subsequent updates to the model fit (default = `TRUE`).
- `update.Rcon` If `TRUE`, the constraint status of variance parameters in the `R.param` list component on termination is updated; this may influence subsequent updates to the model fit (default = `TRUE`).
- `update.step.size` Value for updating shrinkage factor to use in a call to `update()`. If ignored it is set to `0.316` or if `step.size` is explicitly specified on the `update()` call; otherwise the shrinkage factor is set to `update.step.size` in the call to `asreml()` constructed by `update()` (default = `0.316`).
- `use.blas` If `"standard"`, then `asreml` will make use of linear algebra routines compiled into the `asreml` package, otherwise the version of BLAS / LAPACK from R will be used (default = `"standard"`).
- `uspd` If `TRUE`, set the boundary constraint for each parameter in unstructured variance models to `"P"`. Under these conditions, `asreml` checks whether the updated matrix is positive definite; if not, the average information update is replaced with an EM update (see `pxem`) (default = `TRUE`).
- `workspace` Sets the workspace for the core `REML` routines in the form of a number optionally followed directly by a valid measurement unit. Valid units are `kb`, `mb` or `gb`; if no units are given then the value is interpreted as double-precision words (groups of 8 bytes) (default = `"128mb"`).

B.6 Methods and functions

The library `asreml` has an array of methods and functions to fit linear mixed models. Some are required for internal processing, but others help to assist with pre- and post-analyses by preparing data, obtaining relevant matrices, or expanding the results to facilitate interpretation and exploration. Below, a list of the most relevant methods and functions is presented with a brief description. The majority of these require as input an object of class `asreml`, and further details can be found directly in the help from `ASReml-R`.

Table B.2: Relevant methods and functions

Item	Description
`ainverse`	Generates an inverse relationship matrix in sparse triplet form from a pedigree data frame.
`asreml`	Main function to fit the general linear mixed model
`asreml.options`	Set less frequently used `asreml()` options.

Relevant methods and functions

Item	Description
`asreml.read.table`	Reads in a file in table format and creates a data frame that is easily used by `asreml()`.
`chkPed`	Checks a pedigree for consistency, adds missing founders and sorts so that founders appear before individuals.
`coef`	Extract model coefficients (BLUEs and BLUPs) from an `asreml` object.
`fa.init`	Estimate initial parameter values for factor analytic models from the observed variance-covariance matrix.
`lrt`	Generic function to calculate likelihood ratio statistics for fitted models.
`plot`	Four plots are generated: a histogram of the residuals, a Normal Q-Q plot, a plot of residuals against fitted values and a plot of residuals against unit number.
`summary`	A `summary` method for objects inheriting from class `asreml`. It returns a list of several statistics, tables and summaries of the model fit.
`update`	Extract and evaluate the `call` from the fitted object, replacing any arguments with changed values.
`vpredict`	Form functions of variance components from an `asreml` object.
`wald`	Pseudo analysis of variance using incremental Wald statistics or conditional F-tests.

B.7 GLM/GLMM and the exponential family of distributions

B.7.1 Further details on GLMMs

For GLMMs, `asreml()` uses what is commonly referred to as penalized quasi-likelihood or PQL (Breslow and Clayton 1993). The library `asreml()` has implemented the techniques penalized quasi-likelihood or PQL (Breslow and Clayton 1993). The 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). PQL 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 GLIMMIXED 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 (for example, Breslow 2003). As well as the above references, users can consult McCulloch and Searle (2001) for more information about GLMMs.

Most studies investigating PQL have focused 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 to consider with caution. 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).

Appendix C

Variance structures in ASReml-R

C.1 Variance and correlation structure functions

Table C.1 presents the full range of variance models available in ASReml-R with their algebraic descriptions and numbers of parameters. In most cases the algebraic form is for the correlation model (`id()` to `agau()`). However, the models from `diag()` onwards are additional heterogeneous variance models. Recall from Section 4.2 the algebraic forms of the homogeneous and heterogeneous variance models are determined as follows.

Let $\boldsymbol{C}^{(\omega\times\omega)} = [C_{ij}]$ be the correlation matrix for a particular correlation model. If $\boldsymbol{\Sigma}^{(\omega\times\omega)}$ is the corresponding homogeneous variance matrix then

$$\boldsymbol{\Sigma} \quad = \quad \sigma^2\boldsymbol{C}$$

and has just one more parameter than the correlation model. For example, the homogeneous variance model corresponding to the `id()` correlation model has variance matrix $\boldsymbol{\Sigma} = \sigma^2 I_\omega$ (specified `idv()` in the ASReml-R function call, see below) and one parameter. Likewise, if $\boldsymbol{\Sigma}_h^{(\omega\times\omega)}$ is the heterogeneous variance matrix corresponding to $\boldsymbol{C}$, then

$$\boldsymbol{\Sigma}_h \quad = \quad \boldsymbol{DCD}$$

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

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

(specified `idh()` in the `asreml()` call, see following table) and involves the ω parameters $\sigma_1^2 \ldots \sigma_\omega^2$.

Table C.1: Details of the available variance models

function name	description	algebraic form	number of parameters		
			corr	homog. variance	heterog. variance
Correlation models					
`id()`	identity	$C_{ii} = 1,\ C_{ij} = 0,\ \ i \neq j$	0	1	ω
`ar1()`	1^{st} order autoregressive	$C_{ii} = 1,\ C_{i+1,i} = \phi_1$ $C_{ij} = \phi_1 C_{i-1,j},\ i > j+1$ $\lvert\phi_1\rvert < 1$	1	2	$1 + \omega$
`ar2()`	2^{nd} order autoregressive	$C_{ii} = 1,$ $C_{i+1,i} = \phi_1/(1-\phi_2)$ $C_{ij} = \phi_1 C_{i-1,j} + \phi_2 C_{i-2,j},\ i > j+1$ $\lvert\phi_1 \pm \phi_2\rvert < 1,$ $\lvert\phi_1\rvert < 1,\ \lvert\phi_2\rvert < 1$	2	3	$2 + \omega$
`ar3()`	3^{rd} order autoregressive	$C_{ii} = 1,\ \Omega = 1 - \phi_2 - \phi_3(\phi_1 + \phi_3),$ $C_{i+1,i} = (\phi_1 + \phi_2\phi_3)/\Omega,$ $C_{i+2,i} = (\phi_1(\phi_1 + \phi_3) + \phi_2(1-\phi_2))/\Omega,$ $C_{ij} = \phi_1 C_{i-1,j} + \phi_2 C_{i-2,j} + \phi_3 C_{i-3,j},\ i > j+2$ $\lvert\phi_1\rvert < (1-\phi_2), \lvert\phi_2\rvert < 1, \lvert\phi_3\rvert < 1$	3	4	$3 + \omega$
`sar()`	symmetric autoregressive	$C_{ii} = 1,$ $C_{i+1,i} = \phi_1/(1 + \phi_1^2/4)$ $C_{ij} = \phi_1 C_{i-1,j} - \phi_1^2/4\ C_{i-2,j},$ $i > j+1$ $\lvert\phi_1\rvert < 1$	1	2	$1 + \omega$
`sar2()`	constrained autoregressive 3 used for competition	as for AR3 using $\phi_1 = \gamma_1 + 2\gamma_2,$ $\phi_2 = -\gamma_2(2\gamma_1 + \gamma_2),$ $\phi_3 = \gamma_1\gamma_2^2$	2	3	$2 + \omega$
`ma1()`	1^{st} order moving average	$C_{ii} = 1,$ $C_{i+1,i} = -\theta_1/(1 + \theta_1^2)$ $C_{ji} = 0,\ j > i+2$ $\lvert\theta_1\rvert < 1$	1	2	$1 + \omega$

Details of the available variance models

function name	description	algebraic form	number of parameters		
			corr	homog. variance	heterog. variance
`ma2()`	2^{nd} order moving average	$C_{ii} = 1$, $C_{i+1,i} = -\theta_1(1-\theta_2)/(1+\theta_1^2+\theta_2^2)$ $C_{i+2,i} = -\theta_2/(1+\theta_1^2+\theta_2^2)$ $C_{ji} = 0,\ j > i+2$ $\theta_2 \pm \theta_1 < 1$ $\lvert\theta_1\rvert < 1,\ \lvert\theta_2\rvert < 1$	2	3	$2+\omega$
`arma()`	autoregressive moving average	$C_{ii} = 1$, $C_{i+1,i} = (\theta-\phi)(1-\theta\phi)/(1+$ $\theta^2 - 2\theta\phi)$ $C_{ji} = \phi C_{j-1,i},\ j > i+1$ $\lvert\theta\rvert < 1,\ \lvert\phi\rvert < 1$	2	3	$2+\omega$
`cor()`	uniform correlation	$C_{ii} = 1,\ C_{ij} = \theta,\ i \neq j$	1	2	$1+\omega$
`corb()`	banded correlation	$C_{ii} = 1$ $C_{i+j,i} = \phi_j,\ 1 \leq j < \omega$ $\lvert\phi_j\rvert < 1$	j	$j+1$	$j+\omega$
`corg()`	general correlation `corgh()` = `us()`	$C_{ii} = 1$ $C_{ij} = \phi_{ij},\ i \neq j$ $\lvert\phi_{ij}\rvert < 1$	$\frac{\omega(\omega-1)}{2}$	$\frac{\omega(\omega-1)}{2}+1$	$\frac{\omega(\omega-1)}{2}+\omega$
One-dimensional unequally spaced power models					
`exp()`	exponential	$C_{ii} = 1$ $C_{ij} = \phi^{\lvert x_i - x_j\rvert},\ i \neq j$ x_i are *coordinates* $\lvert\phi\rvert < 1$	1	2	$1+\omega$
`gau()`	gaussian	$C_{ii} = 1$ $C_{ij} = \phi^{(x_i-x_j)^2}$ x_i are *coordinates* $\lvert\phi\rvert < 1$	1	2	$1+\omega$
`lvr()`	linear variance	$C_{ij} = (1-\theta_{ij})$ $0 < \phi_1$	1	2	$1+\omega$

Details of the available variance models

function name	description	algebraic form	number of parameters		
			corr	homog. variance	heterog. variance
Two-dimensional irregularly spaced power models					
`iexp()`	isotropic exponential	$C_{ii} = 1$ $C_{ij} = \phi^{\lvert x_i - x_j\rvert + \lvert y_i - y_j\rvert}$ $\boldsymbol{x}$ and (y) vectors of coordinates $\lvert\phi\rvert < 1$	1	2	$1 + \omega$
`igau()`	isotropic gaussian	$C_{ii} = 1$ $C_{ij} = \phi^{(x_i - x_j)^2 + (y_i - y_j)^2}$ $\boldsymbol{x}$ and (y) vectors of coordinates $\lvert\phi\rvert < 1$	1	2	$1 + \omega$
`ieuc()`	isotropic euclidean	$C_{ii} = 1$ $C_{ij} = \phi^{\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}}$ $\boldsymbol{x}$ and (y) vectors of coordinates $\lvert\phi\rvert < 1$	1	2	$1 + \omega$
`sph()`	spherical	$C_{ij} = 1 - \frac{3}{2}\theta_{ij} + \frac{1}{2}\theta_{ij}^3$ $0 < \phi_1$	1	2	$1 + \omega$
`cir()`	circular[†]	$C_{ij} = 1$ $\quad - \frac{2}{\pi}(\theta_{ij}\sqrt{1 - \theta_{ij}^2} + \sin^{-1}\theta_{ij})$ $0 < \phi_1$	1	2	$1 + \omega$
`aexp()`	anisotropic exponential	$C_{ii} = 1$ $C_{ij} = \phi_1^{\lvert x_i - x_j\rvert}\phi_2^{\lvert y_i - y_j\rvert}$ $\boldsymbol{x}$ and (y) vectors of coordinates $\lvert\phi_1\rvert < 1$, $\lvert\phi_2\rvert < 1$	2	3	$2 + \omega$
`agau()`	anisotropic gaussian	$C_{ii} = 1$ $C_{ij} = \phi_1^{(x_i - x_j)^2}\phi_2^{(y_i - y_j)^2}$ $\boldsymbol{x}$ and (y) vectors of coordinates $\lvert\phi_1\rvert < 1$, $\lvert\phi_2\rvert < 1$	2	3	$2 + \omega$
`mtrn()`	Matérn with first $1 \le k \le 5$ parameters specified by the user	C_{ij} =Matérn: see Section C.2.1 $\phi > 0$ range, ν shape(0.5) $\delta > 0$ anisotropy ratio(1), α anisotropy angle(0), $\lambda(1\vert 2)$ metric(2)	k	$k + 1$	$k + \omega$

Details of the available variance models

function name	description	algebraic form	number of parameters		
			corr	homog. variance	heterog. variance
Heterogeneous variance models					
`diag()`	diagonal = `idh()`	$\boldsymbol{\Sigma}_{ii} = \phi_i \; \boldsymbol{\Sigma}_{ij} = 0, \; i \neq j$	-	-	ω
`us()`	unstructured general covariance	$\boldsymbol{\Sigma}_{ij} = \phi_{ij}$	-	-	$\frac{\omega(\omega+1)}{2}$
`ante(, k)`	antedependence order `k` $1 \leq order \leq \omega - 1$	$\boldsymbol{\Sigma}^{-1} = \boldsymbol{UDU}'$ $D_{ii} = d_i, \; D_{ij} = 0, \; i \neq j$ $U_{ii} = 1, \; U_{ij} = u_{ij}, \; 1 \leq j - i \leq$ order $U_{ij} = 0, \; i > j$	-	-	$(k+1)(\omega - k/2)$
`chol(, k)`	cholesky order `k` $1 \leq order \leq \omega - 1$	$\boldsymbol{\Sigma} = \boldsymbol{LDL}'$ $D_{ii} = d_i, \; D_{ij} = 0, \; i \neq j$ $L_{ii} = 1, \; L_{ij} = l_{ij}, \; 1 \leq i-j \leq$ order	-	-	$(k+1)(\omega - k/2)$
`sfa(, k)`	factor analytic order `k`, scaled form	$\boldsymbol{\Sigma} = \boldsymbol{DCD}$, $\boldsymbol{C} = \boldsymbol{FF}' + \boldsymbol{E}$, $\boldsymbol{F}$ *contains correlation factors* $\boldsymbol{E}$ *diagonal* $\boldsymbol{DD} = \text{diag}\,(\boldsymbol{\Sigma})$	-	-	$k\omega + \omega$
`fa(, k)`	factor analytic order `k`, sparse form	$\boldsymbol{\Sigma} = \boldsymbol{\Gamma\Gamma}' + \boldsymbol{\Psi}$, $\boldsymbol{\Gamma}$ *contains covariance factors* $\boldsymbol{\Psi}$ *contains specific variance*	-	-	$k\omega + \omega$
`facv(, k)`	factor analytic order `k`, covariance form	$\boldsymbol{\Sigma} = \boldsymbol{\Gamma\Gamma}' + \boldsymbol{\Psi}$, $\boldsymbol{\Gamma}$ *contains covariance factors* $\boldsymbol{\Psi}$ *contains specific variance*	-	-	$k\omega + \omega$
`rr(, k)`	reduced rank order `k`	$\boldsymbol{\Sigma} = \boldsymbol{\Gamma\Gamma}'$ $\boldsymbol{\Gamma}$ *contains covariance factors*	-	-	$k\omega$
Known variance structures					
`vm()`	known variance model (genetic relationship matrix or its inverse)		-	-	-
General variance models					
`str()`	variance model relating to a sequence of terms in the model		-	-	-
`dsum()`	direct sum structures for the residual error term		-	-	-

†Webster and Oliver (2001)

C.2 Further details on variance structures

C.2.1 The Matérn variance structure `mtrn()`

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

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

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

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

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

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

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

When $\nu = 1.5$ then

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

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

The metric parameter λ is not estimated by ASReml-R; it is usually set to 2 for Euclidean distance. Setting $\lambda = 1$ provides the cityblock metric, which together with $\nu = 0.5$ models a separable AR1×AR1 process. The cityblock metric may be appropriate when the dominant spatial processes

are aligned with rows/columns as occurs in field experiments. Geometric anisotropy is discussed in most geostatistical books (Webster and Oliver 2001; Diggle et al. 2003) but rarely are the anisotropy angle or ratio estimated from the data. Similarly, the smoothness parameter ν is often set *a priori* (Diggle et al. 2003; Kammann and Wand 2003). However, Stein (1999) and Haskard et al. (2007) demonstrate that ν can be reliably estimated even for modest sized data sets, subject to caveats regarding the sampling design.

Estimation

The order of the parameters in `mtrn()`, with their defaults, is ($\phi,\ \nu = 0.5,\ \delta = 1,\ \alpha = 0,\ \lambda = 2$). 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 ϕ estimated and the other parameters set at their defaults then we could use `mtrn(phi = 0.1)` where the starting value for estimation is given as 0.1.

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

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

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

C.2.2 The user-defined variance structure `own()`

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

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

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

```
shf.ar1 <- asreml(
  fixed = yield ~ Variety,
  residual = ~ar1v(Row):ar1(Column),
  data = shf
)
```

We can define `my.ar1` as a regular function:

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

and the following call replaces the intrinsic ASReml-R variance model `ar1()` with the user-defined function `my.ar1()` for the `Column` factor:

```
shf.ar1 <- asreml(
  fixed = yield ~ Variety,
  residual = ~ar1v(Row):own(Column, "my.ar1", 0.1, "R"),
  data = shf
)
```

where the arguments to `own()` are:

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

C.2.3 Other variance structures

- `facv()` An alternative form of the factor analytic model `fa()`. If the size of the matrix modelled is much larger than the number of factors, the sparse or extended formulation `fa()` is usually computationally preferable.
- `vm()` Known variance models. These may be genetic relationship matrices or their inverses if they exist. If an inverse, it must have an `INVERSE` attribute set to `TRUE`. For example, the inverse relationship matrix `harvey.ai` would need to have an inverse attribute set manually:

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

or

```
attr(harvey.ai)$INVERSE <- TRUE
```

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

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

Bibliography

Abramowitz, M., and I. A. Stegun. 1965. Handbook of mathematical functions: With formulas, graphs, and mathematical tables. *National Bureau of Standards Applied Mathematics Series. e.* 55:953.

Breslow, N. E. 2003. *Whither PQL?* UW Biostatistics Working Paper Series, University of Washington. Available online at: https://biostats.bepress.com/uwbiostat/paper192/.

Breslow, N. E., and D. G. Clayton. 1993. Approximate inference in generalized linear mixed models. *Journal of the American Statistical Association.* 88:9–25.

Breslow, N. E., and X. Lin. 1995. Bias correction in generalised linear mixed models with a single component of dispersion. *Biometrika.* 82:81–91.

Cox, D. R., and D. V. Hinkley. 1974. *Theoretical statistics.* Chapman; Hall.

Cox, D. R., and E. J. Snell. 1981. *Applied statistics; principles and examples.* Chapman; Hall.

Cressie, N. A. C. 1991. *Statistics for spatial data.* John Wiley; Sons.

Cullis, B. R., and A. C. Gleeson. 1991. Spatial analysis of field experiments - an extension to two dimensions. *Biometrics.* 47:1449–1460.

Cullis, B. R., A. C. Gleeson, W. J. Lill, J. A. Fisher, and B. J. Read. 1989. A new procedure for the analysis of early generation variety trials. *Applied Statistics.* 38:361–375.

Cullis, B., B. Gogel, A. Verbyla, and R. Thompson. 1998. Spatial analysis of multi-environment early generation variety trials. *Biometrics.* :1–18.

Dempster, A. P., M. R. Selwyn, C. M. Patel, and A. J. Roth. 1984. Statistical and computational aspects of mixed model analysis. *Applied Statistics.* 33:203–214.

Diggle, P. J., P. J. Ribeiro, and O. F. Christensen. 2003. An introduction to model-based geostatistics. P. 43–86 in *Spatial statistics and computational methods*, Moller, J. (ed.). Springer-Verlag.

Draper, N. R., and H. Smith. 1998. *Applied regression analysis.* 3rd ed. John Wyley; Sons, New York.

Gelfand, A. E., M. Fuentes, P. Guttorp, and P. Diggle. 2010. *Handbook of spatial statistics*. Taylor & Francis.

Gilmour, A. R., R. D. Anderson, and A. L. Rae. 1985. The analysis of binomial data by a generalised linear mixed model. *Biometrika*. 72:593–599.

Gilmour, A. R., B. R. Cullis, and A. P. Verbyla. 1997. Accounting for natural and extraneous variation in the analysis of field experiments. *Journal of Agricultural, Biological, and Environmental Statistics*. 2:269–273.

Gilmour, A. R., B. R. Cullis, S. J. Welham, B. J. Gogel, and R. Thompson. 2004. An efficient computing strategy for prediction in mixed linear models. *Computational Statistics and Data Analysis*. 44:571–586.

Gilmour, A. R., B. J. Gogel, B. R. Cullis, S. J. Welham, and R. Thompson. 2002. *ASReml user guide release 1.0*. VSN International, 5 The Waterhouse, Waterhouse St, Hemel Hempstead, HP1 1ES, UK.

Gilmour, A. R., R. Thompson, and B. R. Cullis. 1995. Average information REML: An efficient algorithm for variance parameter estimation in linear mixed models. *Biometrics*. 51:1440–1450.

Gleeson, A. C., and B. R. Cullis. 1987. Residual maximum likelihood (REML) estimation of a neighbour model for field experiments. *Biometrics*. 43:277–288.

Gogel, B. J. 1997. Spatial analysis of multi-environment variety trials/beverley j. gogel. PhD thesis.

Goldstein, H., and J. Rasbash. 1996. Improved approximations for multilevel models with binary response. *Journal of the Royal Statistical Society, Series A*. 159:505–513.

Goldstein, H., J. Rasbash, I. Plewis, D. Draper, W. Browne, M. Yang, G. Woodhouse, and M. Healy. 1998. *A user's guide to MLwiN*. Institute of Education, London. Available online at: http://multilevel.ioe.ac.uk/.

Green, P. J., and B. W. Silverman. 1994. *Nonparametric regression and generalized linear models*. Chapman; Hall.

Harvey, W. R. 1977. *Users' guide to LSML76*. Ohio State University, Columbus.

Harville, D. A., and R. W. Mee. 1984. A mixed model procedure for analysing ordered categorical data. *Biometrics*. 40:393–408.

Haskard, K. A. 2006. Anisotropic Matérn correlation and other issues in model-based geostatistics. PhD thesis, BiometricsSA, University of Adelaide.

Haskard, K. A., B. R. Cullis, and A. P. Verbyla. 2007. Anisotropic Matérn correlation and spatial prediction using REML. *Journal of Agricultural, Biological, and Environmental Statistics*.

12:147–160.

Kammann, E. E., and M. P. Wand. 2003. Geoadditive models. *Applied Statistics*. 52(1):1–18.

Keen, A. 1994. Procedure IRREML. P. Report LWA-94-16 in *GLW-DLO procedure library manual*, Agricultural Mathematics Group, Wageningen, The Netherlands.

Kendall, M. G., and A. Stuart. 1969. *The advanced theory of statistics*. Charles Griffin, London.

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

Lane, P. W., and J. A. Nelder. 1982. Analysis of covariance and standardisation as instances of predicton. *Biometrics*. 38:613–621.

McCullagh, P., and J. A. Nelder. 1994. *Generalized linear models*. 2nd ed. Chapman; Hall, London.

McCulloch, C. E., and S. R. Searle. 2001. *Generalized, linear, and mixed models*. John Wiley; Sons, Inc.

Meuwissen, T. H. E., and Z. Luo. 1992. Computing inbreeding coefficients in large populations. *Genetics Selection Evolution*. 24.

Millar, R. B., and T. J. Willis. 1999. Estimating the relative density of snapper in and around a marine reserve using a log-linear mixed-effects model. *Australian and New Zealand Journal of Statistics*. 41:383–394.

Nelder, J. A. 1977. A reformulation of linear models. *Journal of the Royal Statistical Society, Series A*. 140:48–76.

Nelder, J. A. 1994. The statistics of linear models: Back to basics. *Statistics and Computing*. 4:221–234.

Patterson, H. D., and R. Thompson. 1971. Recovery of interblock information when block sizes are unequal. *Biometrika*. 31:100–109.

Pinheiro, J. C., and D. M. Bates. 2000. *Mixed-effects models in S and S-PLUS*. Springer-Verlaag.

Quaas, R. L., and E. J. Pollak. 1980. Mixed model methodology for farm and ranch beef cattle testing programs. *Journal of Animal Science*. 51(6):1277–1287.

Robinson, G. K. 1991. That BLUP is a good thing: The estimation of random effects. *Statistical Science*. 6:15–51.

Rodriguez, G., and N. Goldman. 2001. Improved estimation procedures for multilevel models with binary response: A case study. *Journal of the Royal Statistical Society, Series A*. 164(2):339–355.

Schall, R. 1991. Estimation in generalized linear models with random effects. *Biometrika*.

78(4):719–27.

Searle, S. R. 1971. *Linear models.* John Wiley; Sons, inc.

Searle, S. R., G. Casella, and C. E. McCulloch. 1992. *Variance components.* J. W. Wiley, New York.

Self, S. C., and K. Y. Liang. 1987. Asymptotic properties of maximum likelihood estimators and likelihood ratio tests under non-standard conditions. *Journal of the American Statistical Society.* 82:605–610.

Stein, M. L. 1999. *Interpolation of spatial data: Some theory for kriging.* Springer-Verlag, New York.

Stevens, M. M., K. M. Fox, G. N. Warren, B. R. Cullis, N. E. Coombes, and L. G. Lewin. 1999. An image analysis technique for assessing resistance in rice cultivars to root-feeding chironomid midge larvae (Diptera: Chironomidae). *Field Crops Research.* 66:25–26.

Thompson, R. 1980. Maximum likelihood estimation of variance components. *Series Statistics.* 11:545–561.

Thompson, R., B. R. Cullis, A. Smith, and A. R. Gilmour. 2003. A sparse implementation of the average information algorithm for factor analytic and reduced rank variance models. *Australian and New Zealand Journal of Statistics.* 45:445–459.

Verbyla, A. P. 1990. A conditional derivation of residual maximum likelihood. *Australian Journal of Statistics.* 32(2):227–230.

Verbyla, A. P., B. R. Cullis, M. G. Kenward, and S. J. Welham. 1999. The analysis of designed experiments and longitudinal data using smoothing splines. *Journal of the Royal Statistical Society, Series C.* :269–311.

Waddington, D., S. J. Welham, A. R. Gilmour, and R. Thompson. 1994. Comparisons of some GLMM estimators for a simple binomial model. *Genstat Newsletter.* 30:13–24.

Webster, R., and M. A. Oliver. 2001. *Geostatistics for environmental scientists.* John Wiley; Sons, Chichester.

Welham, S. J. 2005. GLMM fits a generalized linear mixed model. P. 260–265 in *GenStat reference manual 3: Procedure library PL17,* Payne, R.W., and P.W. Lane (eds.). VSN International, Hemel Hempstead, UK.

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

Welham, S. J., and R. Thompson. 1997. Likelihood ratio tests for fixed model terms using residual maximum likelihood. *Journal of the Royal Statistical Society, Series B.* 59:701–714.

Wolfinger, R. D. 1996. Heterogeneous variance-covariance structures for repeated measures. *Journal of Agricultural, Biological, and Environmental Statistics*. 1:362–389.

Wolfinger, R., and M. O'Connell. 1993. Generalized linear mixed models: A pseudo-likelihood approach. *Journal of Statistical Computation and Simulation*. 48:233–243.

Yates, F. 1935. Complex experiments. *Journal of the Royal Statistical Society, Series B*. 2:181–247.