ASReml
User Guide

Release 4.1

Functional Specification

A R Gilmour
VSN International, Hemel Hempstead, United Kingdom

B J Gogel
University of Adelaide, Australia

B R Cullis
Universtiy of Wollongong, Australia

S J Welham
VSN International, Hemel Hempstead, United Kingdom

R Thompson
Rothamsted Research, Harpenden, United Kingdom

December 22, 2014

ASReml User Guide Release 4.1 Functional Specification

ASReml is a statistical package that fits linear mixed models using Residual Maximum Likelihood (REML). It was a joint venture between the Biometrics Program of NSW Department of Primary Industries and the Biomathematics Unit of Rothamsted Research. Statisticians in Britain and Australia have collaborated in its development.

Main authors:

A. R. Gilmour, B. J. Gogel, B. R. Cullis, S. J. Welham and R. Thompson

Other contributors:

D. Butler, M. Cherry, D. Collins, G. Dutkowski, S. A. Harding, K. Haskard, A. Kelly, S. G. Nielsen, A. Smith, A. P. Verbyla and I. M. S. White.

Author email addresses

arthur.gilmour@Cargovale.com.au
beverley.gogel@adelaide.edu.au
bcullis@uow.edu.au
sue.welham@vsni.co.uk
robin.thompson@rothamsted.ac.uk

Copyright Notice

Published by:

VSN International Ltd,
5 The Waterhouse,
Waterhouse Street,
Hemel Hempstead,
HP1 1ES, UK
email: info@asreml.co.uk
website: http://www.vsni.co.uk/

The correct bibliographical reference for this document is:

Gilmour, A. R., Gogel, B. J., Cullis, B. R., Welham, S. J. and Thompson, R. (2014). ASReml User Guide Release 4.1 Functional Specification, VSN International Ltd, Hemel Hempstead, HP1 1ES, UK www.vsni.co.uk

Preface

ASReml is a statistical package that fits linear mixed models using Residual Maximum Likelihood (REML). It has been under development since 1993 and arose out of collaboration between Arthur Gilmour and Brian Cullis (NSW Department of Primary Industries) and Robin Thompson and Sue Welham (Rothamsted Research) to research into the analysis of mixed models and to develop appropriate software, building on their wide expertise in relevant areas including the development of methods that are both statistically and computationally efficient, the analysis of animal and plant breeding data, the analysis of spatial and longitudinal data and the production of widely used statistical software. More recently, VSN International acquired the right to ASReml from these sponsoring organizations and now directly supports Arthur Gilmour and Sue Welham for further computational developments and research on the analysis of mixed models. Release 4 of ASReml was first distributed in 2014. A major enhancement in this release is the introduction of an alternative, functional, specification of linear mixed models. For the convenience of users, three documents have been prepared, *i.* a guide to Release 4 using the original, still supported, model specification, *ii.* this document which is a guide using the new functional model specification and *iii* a document ASReml Update: What's new in Release 4, which highlights the changes from Release 3.

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

ASReml provides a stable platform for delivering well established procedures while also delivering current research in the application of linear mixed models. The strength of ASReml is the use of the Average Information (AI) algorithm and sparse matrix methods for fiting the linear mixed model. This enables it to analyse large and complex data sets quite efficiently.

One of the strengths of ASReml is the wide range of variance models for the random effects in the linear mixed model that are available. There is a potential cost for this wide choice. 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 using 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 in the context of real analyses but also to indicate some of the modelling approaches we have found useful.

There are several interfaces to the core functionality of ASReml. The program name ASReml relates to the primary program. ASReml-W refers to the user interface program developed by VSN and distributed with ASReml. ASReml-R refers to the S language interface to a DLL of the core ASReml routines. GenStat uses the same core routines for its REML directive. Both of these have good data manipulation and graphical facilities.

The focus in developing ASReml has been on the core engine and it is freely acknowledged that its user interface is not to the level of these other packages. Nevertheless, as the developer's interface, it is functional, it gives access to everything that the core can do and is especially suited to batch processing and running of large models without the overheads of other systems.

This guide has 15 chapters. Chapter 1 introduces ASReml and describes the conventions used in this guide. Chapter 2 outlines some basic theory while Chapter 3 presents an overview of the syntax of ASReml through a simple example. Data file preparation is described in Chapter 4 and Chapter 5 describes how to input data into ASReml. Chapters 6 and 7 are key chapters which present the syntax for specifying the linear model and the variance models for the random effects in the linear mixed model. Chapters 8 and 9 describe special commands for multivariate and genetic analyses respectively. Chapter 10 deals with prediction of linear functions of fixed and random effects in the linear mixed model, Chapter 11 demonstrates running an ASReml job, Chapter 12 describes the merging of data files and Chapter 13 presents the syntax for forming functions of variance components. Chapter 14 gives a detailed explanation of the output files. Chapter 15 gives an overview of the error messages generated in ASReml and some guidance as to their probable cause. The guide concludes with the most extensive chapter which presents the analysis of a range of data examples.

In brief, the improvements in Release 4 include developments associated with input include generating initial values, generating a template to allow an alternative way of presenting parametric information associated with variance structures, new facilities for reading in data files and defining factor names and improved facilities for reading relationship matrices and better explanation of a simpler way of constructing variances of functions of parameters. Among the developments associated with analysis are making it easier to specify functions of variance parameters using names rather than numbers, fitting factor effects with large random regression models, such as commonly used with marker data, fitting linear relationships among variance structure parameters and calculating information criteria. The developments associated with output include writing out design matrices. A major development in Release 4 is an alternative model specification using a functional approach. Prior to Release 4 a structural specification was used in which variance models were applied by imposing variance structures on random model terms and/or the residual error term after the mixed model had been specified. In this case, the variance models were presented in a separate part of the input file. The functional specification offers an alternative to the structural specfication in which the variance structures for random model terms and the residual error term are specified in the linear mixed model definition by wrapping terms with the required variance model function. This approach is more concise, less error-prone and more automatic for specifying multi-section residual variances.

The data sets and ASReml input used in this guide are available from `http://www.vsni.co.uk/products/asreml` as well as in the examples directory created under the standard installation. They remain the property of the authors or of the original source but may be freely distributed provided the source is acknowledged. The authors would appreciate feedback and suggestions for improvements to the program and this guide. Proceeds from the licensing of ASReml are used to support continued development to implement new developments in the application of linear mixed models. The developmental version is available to supported licensees via a website upon request to VSN. Most users will not need to access the developmental version unless they are actively involved in testing a new development.

Acknowledgements

We gratefully acknowledge the Grains Research and Development Corporation of Australia for their financial support for our research since 1988. Brian Cullis and Arthur Gilmour wish to thank the NSW Department of Primary Industries, and more recently the University of Wollongong, for providing a stimulating and exciting environment for applied biometrical research and consulting. Rothamsted Research receives grant-aided support from the Biotechnology and Biological Sciences Research Council of the United Kingdom. We sincerely thank Ari Verbyla, Dave Butler and Alison Smith, the other members of the ASReml 'team'. Ari contributed the cubic smoothing splines technology, information for the Marker map imputation, on-going testing of the software and numerous helpful discussions and insight. Dave Butler has developed the ASReml-R package. Alison contributed to the development of many of the approaches for the analysis of multi-section trials. We also thank Ian White for his contribution to the spline methodology, and Simon Harding for the licensing and installation software and for his development of the user interface program ASReml-W. The Matérn function material was developed with Kathy Haskard and Brian Cullis, and the denominator degrees of freedom material was developed with Sharon Nielsen, a Masters student with Brian Cullis. Damian Collins contributed the `PREDICT !PLOT` material. Greg Dutkowski has contributed to the extended pedigree options. The `asremload.dll` functionality is provided under license to VSN. Alison Kelly has helped with the review of the XFA models. Finally, we especially thank our close associates who continually test the enhancements. Arthur Gilmour acknowledges the grace of God through Jesus Christ our Saviour. *In Him are hidden all the treasures of wisdom and knowledge.* Colossians 2:3.

Contents

List of Tables

List of Figures

1 Introduction

1.1 What ASReml can do

ASReml (pronounced *A S Rem el*) is used to fit linear mixed models to quite large data sets with complex variance models. It extends the range of variance models available for the analysis of experimental data. ASReml has application in the analysis of

- (un)balanced longitudinal data,
- repeated measures data (multivariate analysis of variance and spline type models),
- (un)balanced designed experiments,
- multi-environment trials and meta analysis,
- univariate and multivariate animal breeding and genetics data (involving a relationship matrix for correlated effects),
- regular or irregular spatial data.

The engine of ASReml underpins the REML procedure in GENSTAT. An interface for R called ASReml-R is available and runs under the same license as the ASReml program. While these interfaces will be adequate for many analyses, some large problems will need to use ASReml. The ASReml user interface is terse. Most effort has been directed towards efficiency of the engine. It normally operates in a batch mode.

Problem size depends on the sparsity of the mixed model equations and the size of your computer. However, models with 500,000 effects have been fitted successfully. The computational efficiency of ASReml arises from using the Average Information REML procedure (giving quadratic convergence) and sparse matrix operations. ASReml has been operational since March 1996 and is updated periodically.

1.2 Installation

Installation instructions are distributed with the program. If you require help with installation or licensing, please email `support@asreml.co.uk`.

1.3 User Interface

ASReml is essentially a batch program with some optional interactive features. The typical sequence of operations when using ASReml is

- Prepare the data (typically using a spreadsheet or data base program)
- Export that data as an ASCII file (for example export it as a `.csv` (comma separated values) file from Excel)
- Prepare a job file with filename extension `.as`
- Run the job file with ASReml
- Review the various output files
- revise the job and re run it, or
- extract pertinent results for your report.

You need an ASCII editor to prepare input files and review and print output files. Two commonly used editors are:

1.3.1 ASReml-W

The ASReml-W interface is a graphical tool allowing the user to edit programs, run and then view the output, before saving results. It is available on the following platforms:

- Windows (32-bit and 64-bit),
- Linux (32-bit and 64-bit, various incantations),

ASReml-W has a built-in help system explaining its use.

1.3.2 ConTEXT

ConTEXT is a third-party freeware text editor, with programming extensions which make it a suitable environment for running ASReml under Windows. The ConTEXT directory on the CD-ROM includes installation files and instructions for configuring it for use in ASReml. Full details of ConTEXT are available from `http://www.contexteditor.org/`.

1.4 How to use this guide

The guide consists of 16 chapters. Chapter 1 introduces ASReml and describes the conventions used in the guide. Chapter 2 outlines some basic theory which you may need to come back to.

New ASReml users are advised to read Chapter 3 before attempting to code their first job. It presents an overview of basic ASReml coding demonstrated on a real data example. Chapter 16 presents a range of examples to assist users further. When coding your first job, look for an example to use as a model.

Data file preparationis described in Chapter 4, and Chapter 5 describes how to input data into ASReml. Chapters 6 and 7 are key chapters which present the syntax for specifying the linear model and the variance models for the random effects in the linear mixed model.Variance modelling is a complex aspect of analysis. We introduce variance modelling in ASReml by example in Chapter 16.

Chapters 8 and 9 describe special commands for multivariate and genetic analyses respectively. Chapter 10 deals with prediction offixed and random effects from the linear mixed model and Chapter 13 presents the syntax for forming functions of variance components such as heritability.

Chapter 11 discusses the operating system level command for running an ASReml job. Chapter 12 describes a new data merging facility. Chapter 14 gives a detailed explanationof the output files. Chapter 15 gives an overview of the error messages generated in ASReml and some guidance as to their probable cause.

1.5 Getting assistance and the ASReml forum

The ASReml help accessable through ASReml-W can also be linked to ConText or accessed directly (`ASReml.chm`).

There is a User Area on the website (`http://www.VSNi.co.uk` select `ASReml` and then `User Area`) which contains contributed material that may be of assistance.

Users with a support contract with VSN should email `support@asreml.co.uk` for assistance with installation and running ASReml. When requesting help, please send the input command file, the data file and the corresponding primary output file along with a description of the problem. All ASReml users (including unsupported users) are encouraged to join the ASReml forum; register now at `http://www.vsni.co.uk/forum`.

If ASReml appears to be failing, then please send details of the problem to `support@vsni.co.uk`.

1.6 Typographic conventions

A hands on approach is the best way to develop a working understanding of a new computing package. We therefore begin by presenting a guided tour of ASReml using a sample data set for demonstration (see Chapter 3). Throughout the guide new concepts are demonstrated by example wherever possible.

In this guide you will find framed sample boxes to the right of the page as shown here. These contain ASReml command file (sample) code. Note that

- the code under discussion is highlighted in bold type for easy identification,
- the continuation symbol (⋮) is used to indicate that some of the original code is omitted.

```
An example ASReml code box
bold type highlights sections of code
currently under discussion
remaining code is not highlighted
⋮ indicates that some of the original
code is omitted from the display
```

Data examples are displayed in larger boxes in the body of the text, see, for example, page 40. Other conventions are as follows:

- keyboard key names appear in SMALLCAPS, for example, TAB and ESC,
- example code within the body of the text is in `this size and font` and is highlighted in bold type, see pages 33 and 49,
- in the presentation of general ASReml syntax, for example

  [*path*] `asreml` *basename*[`.as`] [*arguments*]
  - `typewriter font` is used for text that must be typed verbatim, for example, `asreml` and `.as` after *basename* in the example,
  - *italic font* is used to name information to be supplied by the user, for example, *basename* stands for the name of a file with an `.as` filename extension,
  - square brackets indicate that the enclosed text and/or arguments are not always required. Do not enter these square brackets.
- ASReml output is in this size and font, see page 34,
- `this font` is used for all other code.

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

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

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

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

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

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

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

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

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

2.1.2 Partitioning the fixed and random model terms

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

2.1.3 G structure for the random model terms

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

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

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 $\operatorname{var}(\boldsymbol{u}_i) = \boldsymbol{G}_i$ for $i = 1 \ldots b$). 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 $b' < b$). In some other (less likely but possible) cases, we may wish to separate one model term over several independent parts (then $b' > b$), see Section 7.2.1.

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

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

and

$$\mathrm{var}\,(\boldsymbol{e}) \;=\; \sigma_e^2 \boldsymbol{I}_n$$

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

$$\mathrm{var}\,(\boldsymbol{y}) \;=\; \sum_{i=1}^{b} \sigma_{u_i}^2 \boldsymbol{Z}_i \boldsymbol{Z}_i^{\mathsf{T}} + \sigma_e^2 \boldsymbol{I}_n. \tag{2.5}$$

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

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

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

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

Typically a variance structure is specified for each random model term and often more complex models than the simple IID model are specified. ASReml offers a wide range of variance models to choose from. A full listing is in Table 7.6 and details are provided in Chapter 7.

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,* see above. Other parameterizations are possible and are sometimes useful. For example, in some of the early development of REML for the traditional mixed model of (2.5), the variance matrix was parameterized as the equivalent model

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

for γ_{g_i} being the ratio of the variance component for the random term $\boldsymbol{u}_i$ relative to error variance, that is, $\gamma_{g_i} = \sigma^2_{u_i}/\sigma^2_e$. In this case ASReml 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

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

which we will refer to as the *gamma parameterization*, and the individual variance structure parameters in $\boldsymbol{\gamma}_g$ and $\boldsymbol{\gamma}_r$ will be referred to as *gammas*. ASReml switches between the sigma and gamma parameterizations for estimation. This is discussed in Section 7.6.

2.1.7 Parameter types

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

2.1.8 Variance structures for the random model terms

The random model terms $\boldsymbol{u}_i$ in $\boldsymbol{u}$ define the random effects and associated design matrices, $\boldsymbol{Z}_i \in \boldsymbol{Z}$, but additional information is required before the model can be fitted. This extra step involves defining the G structure for each term. In Release 4, this is achieved by using functions to directly apply variance models to the individual component factors in a random model term to define $\boldsymbol{G}_i$. This produces a consolidated model term that simultaneously defines both the design matrix ($\boldsymbol{Z}_i$) and variance model ($\boldsymbol{G}_i$). This process is described in detail in Chapter 7 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 m and n levels respectively and the "." operator forms a term with levels corresponding to the combinations of all levels of `A` with all levels of `B`. The effects ab_{ij} for `A.B` are generated with the levels of `B` nested in the levels of `A`, ie. the levels of `B` cycling fastest:

$$\boldsymbol{(ab)} = (ab_{11},\ ab_{12}, \ldots ab_{1n}, ab_{21},\ ab_{22}, \ldots ab_{2n}, \ldots, ab_{m1},\ ab_{m2}, \ldots ab_{mn})^{\mathsf{T}}$$

Now consider the variance model for the term `A.B`. If we specify our variance model generically as

```
vmodel1(A).vmodel2(B)
```

where `vmodel1` is a variance model function with variance matrix $\boldsymbol{A} = [A_{ij}]$ and `vmodel2` is a variance model function with variance matrix $\boldsymbol{B} = [B_{kl}]$, then the G structure for this term is defined by

$$\text{cov}\,(ab_{ik}, ab_{jl}) = A_{ij} \times B_{kl}. \tag{2.9}$$

This means that the covariance between two effects ab_{ik} and ab_{jl} in $\boldsymbol{(ab)}$ is constructed as the product of the covariance between a_i and a_j in model $\boldsymbol{A}$ i.e. its $(i,j)^{th}$ element A_{ij}, and the covariance between b_k and b_l in model $\boldsymbol{B}$ i.e. its $(k,l)^{th}$ element B_{kl}.

Example 2.3 A simple direct product structure

If `A` has 3 levels and `B` has 2 levels, then the term `A.B` would have the 6 levels:

$$\boldsymbol{(ab)} = (ab_{11},\ ab_{12},\ ab_{21},\ ab_{22},\ ab_{31},\ ab_{32})^{\mathsf{T}}.$$

Using magenta and blue to highlight terms associated with `A` and `B` respectively in cov (ab_{21}, ab_{32}) , if

$$\texttt{var(A)} = \begin{bmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \end{bmatrix} \text{ and } \texttt{var(B)} = \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix} \text{ then}$$

$$\text{cov}\,(ab_{21}, ab_{32}) = A_{23} \times B_{12}.$$

2.1.10 Direct product structures

Mathematically, the result (2.9) is known as a *direct product structure* and is written in full as

$$\begin{aligned}\operatorname{var}((\boldsymbol{ab})) &= \boldsymbol{A}\otimes\boldsymbol{B}\\ &= \begin{bmatrix} A_{11}\boldsymbol{B} & \dots & A_{1p}\boldsymbol{B}\\ \vdots & \ddots & \vdots\\ A_{m1}\boldsymbol{B} & \ddots & A_{mp}\boldsymbol{B}\end{bmatrix}.\end{aligned}$$

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

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

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

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

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

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

where $\boldsymbol{\Sigma}^{(n_t\times n_t)}$ is a general or *unstructured* variance matrix. See Chapter 7 for details on specifying separable R structures in ASReml.

2.1.12 Direct products in G structures

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

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

where $\boldsymbol{\Sigma}$ is the variance matrix for sites. This would imply that the varieties are independent random effects within each site, have different variances at each site, and are correlated across sites. **Important** Whenever a random term is formed as the interaction of two factors you should consider whether the IID assumption is sufficient or if a direct product structure might be more appropriate. See Chapter 7 for details on specifying separable G structures in ASReml.

2.1.13 Range of variance models for R and G structures

A range of models are available for the components of both R and G structures. They include correlation (C) models (that is, where the diagonals are 1), or covariance (V) models and are discussed in detail in Chapter 7. Among the range of correlation models are:

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

Among the range of covariance models are:

- scaled identity (that is, independent and identically distributed with homogenous variances)
- diagonal (that is, independent with heterogeneous variances)
- antedependence

- unstructured
- factor analytic.

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

2.1.14 Combining variance models in R and G structures

The combination of variance models in separable G and R structures is a difficult and important concept. This is discussed in detail in Chapter 7.

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. 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

$$\boldsymbol{y} \sim N(\boldsymbol{X\tau},\ \boldsymbol{H}), \tag{2.10}$$

where $\boldsymbol{H} = \boldsymbol{ZG}(\boldsymbol{\sigma}_g)\boldsymbol{Z}^\mathsf{T} + \boldsymbol{R}_v(\boldsymbol{\sigma}_r)$. REML does not use (2.10) 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

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

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

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

The full distribution of $\boldsymbol{L}^\mathsf{T}\boldsymbol{y}$ 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,

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

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

$$\begin{aligned} \ell_R &= -\frac{1}{2}(\log\det \boldsymbol{L}_2^\mathsf{T}\boldsymbol{H}^{-1}\boldsymbol{L}_2 + \boldsymbol{y}_2^\mathsf{T}(\boldsymbol{L}_2^\mathsf{T}\boldsymbol{H}\boldsymbol{L}_2)^{-1}\boldsymbol{y}_2) \\ &= -\frac{1}{2}(\log\det \boldsymbol{X}^\mathsf{T}\boldsymbol{H}^{-1}\boldsymbol{X} + \log\det \boldsymbol{H} + \boldsymbol{y}^\mathsf{T}\boldsymbol{P}\boldsymbol{y}_2) \end{aligned} \tag{2.11}$$

where

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

Note that $\boldsymbol{y}^\mathsf{T}\boldsymbol{P}\boldsymbol{y} = (\boldsymbol{y} - \boldsymbol{X}\hat{\boldsymbol{\tau}})^\mathsf{T}\boldsymbol{H}^{-1}(\boldsymbol{y} - \boldsymbol{X}\hat{\boldsymbol{\tau}})$. The log-likelihood (2.11) depends on $\boldsymbol{X}$ and not on the particular non-unique transformation defined by $\boldsymbol{L}$.

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

$$\ell_R = -\frac{1}{2}(\log\det \boldsymbol{C} + \log\det \boldsymbol{R}_v + \log\det \boldsymbol{G} + \boldsymbol{y}^\mathsf{T}\boldsymbol{P}\boldsymbol{y}). \tag{2.12}$$

We can also write

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

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

$$U(\kappa_i) = \partial\ell_R/\partial\kappa_i = -\frac{1}{2}[\mathrm{tr}\,(\boldsymbol{P}\boldsymbol{H}_i) - \boldsymbol{y}^\mathsf{T}\boldsymbol{P}\boldsymbol{H}_i\boldsymbol{P}\boldsymbol{y}] \tag{2.13}$$

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

The elements of the observed information matrix are

$$\begin{aligned} -\frac{\partial^2\ell_R}{\partial\kappa_i\partial\kappa_j} &= \frac{1}{2}\mathrm{tr}\,(\boldsymbol{P}\boldsymbol{H}_{ij}) - \frac{1}{2}\mathrm{tr}\,(\boldsymbol{P}\boldsymbol{H}_i\boldsymbol{P}\boldsymbol{H}_j) \\ &\qquad + \boldsymbol{y}^\mathsf{T}\boldsymbol{P}\boldsymbol{H}_i\boldsymbol{P}\boldsymbol{H}_j\boldsymbol{P}\boldsymbol{y} - \frac{1}{2}\boldsymbol{y}^\mathsf{T}\boldsymbol{P}\boldsymbol{H}_{ij}\boldsymbol{P}\boldsymbol{y} \end{aligned} \tag{2.14}$$

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

The elements of the expected information matrix are

$$\mathrm{E}\left(-\frac{\partial^2\ell_R}{\partial\kappa_i\partial\kappa_j}\right) = \frac{1}{2}\mathrm{tr}\,(\boldsymbol{P}\boldsymbol{H}_i\boldsymbol{P}\boldsymbol{H}_j)\,. \tag{2.15}$$

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

$$\boldsymbol{\kappa}^{(1)} = \boldsymbol{\kappa}^{(0)} + \boldsymbol{I}(\boldsymbol{\kappa}^{(0)}, \boldsymbol{\kappa}^{(0)})^{-1}\boldsymbol{U}(\boldsymbol{\kappa}^{(0)}) \tag{2.16}$$

where $\boldsymbol{U}(\boldsymbol{\kappa}^{(0)})$ is the score vector (2.13) and $\boldsymbol{I}(\boldsymbol{\kappa}^{(\mathbf{0})}, \boldsymbol{\kappa}^{(\mathbf{0})})$ is the expected information matrix (2.15) of $\boldsymbol{\kappa}$ evaluated at $\boldsymbol{\kappa}^{(0)}$.

For large models or large data sets, the evaluation of the trace terms in either (2.14) or (2.15) is either not feasible or is very computer intensive. To overcome this problem ASReml uses the AI algorithm (Gilmour, Thompson and Cullis, 1995). The matrix denoted by $\mathcal{I}_A$ is obtained by averaging (2.14) and (2.15) and approximating $\boldsymbol{y}^\mathsf{T}\boldsymbol{P}\boldsymbol{H}_{ij}\boldsymbol{P}\boldsymbol{y}$ by its expectation, $\operatorname{tr}(\boldsymbol{P}\boldsymbol{H}_{ij})$ in those cases when $\boldsymbol{H}_{ij} \neq 0$. For variance components models (that is those linear with respect to variances in $\boldsymbol{H}$), the terms in $\mathcal{I}_A$ are exact averages of those in (2.14) and (2.15). The basic idea is to use $\mathcal{I}_A(\kappa_i, \kappa_j)$ in place of the expected information matrix in (2.16) to update $\boldsymbol{\kappa}$.

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

$$\mathcal{I}_A(\kappa_i, \kappa_j) \quad = \quad \frac{1}{2}\boldsymbol{y}^\mathsf{T}\boldsymbol{P}\boldsymbol{H}_i\boldsymbol{P}\boldsymbol{H}_j\boldsymbol{P}\boldsymbol{y}. \tag{2.17}$$

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 (2.18). In this form the AI matrix is relatively straightforward to calculate.

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

2.2.2 Estimation/prediction of the fixed and random effects

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

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

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

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

$$\begin{bmatrix} \boldsymbol{X}^\mathsf{T}\boldsymbol{R}_v^{-1}\boldsymbol{X} & \boldsymbol{X}^\mathsf{T}\boldsymbol{R}_v^{-1}\boldsymbol{Z} \\ \boldsymbol{Z}^\mathsf{T}\boldsymbol{R}_v^{-1}\boldsymbol{X} & \boldsymbol{Z}^\mathsf{T}\boldsymbol{R}_v^{-1}\boldsymbol{Z} + \boldsymbol{G}^{-1} \end{bmatrix} \begin{bmatrix} \hat{\boldsymbol{\tau}} \\ \tilde{\boldsymbol{u}} \end{bmatrix} = \begin{bmatrix} \boldsymbol{X}^\mathsf{T}\boldsymbol{R}_v^{-1}\boldsymbol{y} \\ \boldsymbol{Z}^\mathsf{T}\boldsymbol{R}_v^{-1}\boldsymbol{y} \end{bmatrix}. \tag{2.18}$$

These can be written as

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

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

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

The solution of (2.18) 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 best linear unbiased estimator (BLUE) of $\boldsymbol{\tau}$, while $\tilde{\boldsymbol{u}}$ is the best linear unbiased predictor (BLUP) of $\boldsymbol{u}$ for known $\boldsymbol{\sigma}_g$ and $\boldsymbol{\sigma}_r$. We also note that

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

2.2.3 Use of the gamma parameterization

ASReml uses either the gamma or sigma parameterization for estimation depending on the residual specification. The current default for univariate, single section data-sets is the gamma parameterization. In this case, all scale parameters are estimated as a ratio with respect to the residual variance, σ_e^2, and any parameters that measure only correlation are unchanged. See Chapter 7 for more detail.

2.3 What are BLUPs?

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

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

where $\bar{\boldsymbol{y}}$ is the vector of treatment means, $\bar{y}_{..}$ 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 $\boldsymbol{A\psi}$. If $\boldsymbol{\psi} = \mathbf{0}$, the BLUP in (2.19) becomes

$$\tilde{\boldsymbol{u}} = \frac{r\sigma_b^2}{r\sigma_b^2 + \sigma^2}(\bar{\boldsymbol{y}} - \mathbf{1}\bar{y}_{..}) \tag{2.20}$$

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 (2.20) represents a weighted mean which involves the prior assumption that the u_i have zero mean.

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

2.4 Inference: Random effects

2.4.1 Tests of hypotheses: variance parameters

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

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

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

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, then the asymptotic distribution of the REMLRT, under the restricted model is $\chi^2_{r_2-r_1}$.

The REMLRT is implicitly two-sided, and must be adjusted when the test involves an hypothesis with the parameter on the boundary of the parameter space. It can be shown that for a single variance component, the theoretical asymptotic distribution of the REMLRT is a mixture of χ^2 variates, where the mixing probabilities are 0.5, one with 0 degrees of freedom (spike at 0) and the other with 1 degree of freedom. The approximate P-value for the REMLRT statistic (D), is 0.5(1-Pr($\chi^2_1 \leq d$)) where d is the observed value of D. This has a 5% critical value of 2.71 in contrast to the 3.84 critical value for a χ^2 variate with 1 degree of freedom. The distribution of the REMLRT for the test that k variance components are zero, or tests involved in random regressions, which involve both variance and covariance components, involves a mixture of χ^2 variates from 0 to k degrees of freedom. See Self and Liang (1987) for details.

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

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

$$\begin{aligned}\mathsf{AIC} &= -2\ell_{Ri} + 2t_i \\ \mathsf{BIC} &= -2\ell_{Ri} + t_i \log \nu \qquad (2.22)\end{aligned}$$

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

2.4.2 Diagnostics

In this section we will briefly review some of the diagnostics that have been implemented in ASReml 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 BLUP of the residual vector is given by

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

It follows that

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

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

The `!OUTLIER` qualifier invokes a partial implementation of research by Alison Smith, Ari Verbyla and Brian Cullis. With this qualifier, ASReml writes

- $\boldsymbol{G}^{-1}\boldsymbol{u}$ and $\boldsymbol{G}^{-1}\boldsymbol{u}/\mathrm{diag}\sqrt{\boldsymbol{G}^{-1} - \boldsymbol{G}^{-1}\boldsymbol{C}^{ZZ}\boldsymbol{G}^{-1}}$ to the `.sln` file,
- $\boldsymbol{R}_v^{-1}\boldsymbol{e}$ and $\boldsymbol{R}_v^{-1}\boldsymbol{e}/\mathrm{diag}\sqrt{\boldsymbol{R}_v^{-1} - \boldsymbol{R}_v^{-1}\boldsymbol{W}\boldsymbol{C}^{-1}\boldsymbol{W}^{\mathsf{T}}\boldsymbol{R}_v^{-1}}$ to the `.yht` file,
- and copies lines where the last ratio exceeds 3 in magnitude to the `.res` file
- and reports the number of such lines to the `.asr` file.
- It has not been validated for multivariate models or XFA models with zero Ψs.

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 reported by ASReml has two forms depending on whether the spatial coordinates represent a complete rectangular lattice (as typical of a field trial) or not. In the lattice case, 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 displacements. As there will be many v_{ij} with the same displacements, ASReml calculates the means for each displacement pair l_{ij1}, l_{ij2} either ignoring the signs (default) or separately for same sign and opposite sign (`!TWOWAY`), after grouping the larger displacements: 9-10, 11-14, 15-20, ... The result is displayed as a perspective plot (see page 236) of the one or two surfaces indexed by absolute displacement group. In this case, the two directions may be on different scales.

Otherwise ASReml forms a variogram based on polar coordinates. It calculates the distance between points $d_{ij} = \sqrt{l_{ij1}^2 + l_{ij2}^2}$ and an angle θ_{ij} $(-180 < \theta_{ij} < 180)$ subtended by the line from $(0, 0)$ to (l_{ij1}, l_{ij2}) with the x-axis. The angle can be calculated as $\theta_{ij} = \tan^{-1}(l_{ij1}/l_{ij2})$ choosing $(0 < \theta_{ij} < 180)$ if $l_{ij2} > 0$ and $(-180 < \theta_{ij} < 0)$ if $l_{ij2} < 0$. Note that the variogram has angular symmetry in that $v_{ij} = v_{ji}$, $d_{ij} = d_{ji}$ and $|\theta_{ij} - \theta_{ji}| = 180$. The variogram presented averages the v_{ij} within 12 distance classes and 4, 6 or 8 sectors (selected using a `!VGSECTORS` qualifier) centred on an angle of $(i-1)*180/s$ $(i = 1, ...s)$. A figure is produced which reports the trends in $\bar{v}_{ij}$ with increasing distance for each sector.

ASReml also computes the variogram from predictors of random effects which appear to have a variance structures defined in terms of distance. The variogram details are reported in the `.res` file.

2.5 Inference: Fixed effects

2.5.1 Introduction

Inference for fixed effects in linear mixed models introduces some difficulties. In general, the methods used to construct F-tests in analysis of variance and regression cannot be used for the diversity of applications of the general linear mixed model available in ASReml. One approach would be to use likelihood ratio methods (see 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

$$\mathcal{W} = (\boldsymbol{L}\hat{\boldsymbol{\tau}} - \boldsymbol{l})^{\mathsf{T}}\{\boldsymbol{L}(\boldsymbol{X}^{\mathsf{T}}\boldsymbol{H}^{-1}\boldsymbol{X})^{-1}\boldsymbol{L}^{\mathsf{T}}\}^{-1}(\boldsymbol{L}\hat{\boldsymbol{\tau}} - \boldsymbol{l}) \tag{2.24}$$

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

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

In the following we describe the facilities now available in ASReml for conducting inference concerning terms which are the in 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 F statistics and partial implementation of the Kenward and Roger adjustments.

2.5.2 Incremental and conditional Wald F Statistics

The basic tool for inference is the Wald statistic defined in equation 2.17. ASReml produces a test of fixed effects, that reduces to an F statistic in special cases, by dividing the Wald statistic, constructed with $\boldsymbol{l} = 0$, 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. 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. These Wald F statistics are labelled `F-inc` (for incremental) and `F-con` (for conditional) respectively. For balanced designs, these Wald F statistics are numerically identical to the F statistics obtained from the standard analysis of variance.

The first method for computing Wald statistics (for each term) is the so-called "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

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

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

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

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

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

and

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

Implicit in these calculations is that

- we only compute Wald statistics for *estimable* functions (Searle, 1971, page 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

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

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 Nelder defines marginality for terms in a factorial linear model with qualitative factors, but later Nelder (1994) extended this concept to functional marginality for terms involving quantitative covariates and for mixed terms which involve an interaction between quantitative covariates and qualitative factors. Referring to our simple illustrative example above, with a full factorial linear model given symbolically by

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

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

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

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

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

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

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

The preceeding table includes a so-called `M` (marginality) code reported by ASReml when conditional Wald statistics are presented. All terms with the highest `M` code letter are tested conditionally on all other terms in the model, i.e. by dropping the term from the maximum model. All terms with the preceeding `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 its test is conditional on all covariates but no factors. Following is some ASReml output from the `.aov` file which reports the terms in the conditional statistics.

```
                    Marginality pattern for F-con calculation
                            -- Model terms --
 Model Term           DF   1  2  3  4  5  6  7  8

  1 mu                 1   *  .  .  .  .  .  .  .
  2 water              1   I  *  C  C  .  .  c  .
  3 variety            7   I  I  *  C  .  c  .  .
  4 sow                2   I  I  I  *  C  .  .  .
  5 water.variety      7   I  I  I  I  *  C  C  .

  6 water.sow          2   I  I  I  I  I  *  C  .
  7 variety.sow       14   I  I  I  I  I  I  *  .
  8 water.variety.sow 14   I  I  I  I  I  I  I  *
```

`F-inc` tests the additional variation explained when the term (`*`) is added to a model consisting of the `I` terms. `F-con` tests the additional variation explained when the term (`*`) is added to a model consisting of the `I` and `C/c` terms. Any `c` terms are ignored in calculating DenDF for `F-con` using *numerical* derivatives for computational reasons. The `.` terms are ignored for both `F-inc` and `F-con` tests.

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 F statistics will be the same. However, it is not uncommon for this model to be presented to ASReml as

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

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

When using the conditional Wald F statistic, it is important to know what the "maximal conditional" model (MCM) is for that particular statistic. It is given explicitly in the `.aov` file. The purpose of the conditional Wald F statistic is to facilitate inference for fixed effects. It is not meant to be prescriptive of the appropriate test nor is the algorithm for determining the MCM foolproof.

The Wald statistics are collectively presented in a summary table in the `.asr` file. The basic table includes the numerator degrees of freedom (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 `!FCON` is specified. A conditional Wald F statistic is not reported for `mu` in the `.asr` but is in the `.aov` file (adjusted for covariates).

The `!FOWN` qualifier (page 78) allows the user to replace any/all of the conditional Wald F statistics with tests of the same terms but adjusted for other model terms as specified by the user; the `!FOWN` test is not performed if it implies a change in degrees of freedom from that obtained by the incremental model.

2.5.3 Kenward and Roger adjustments

In moderately sized analyses, ASReml will also include the denominator degrees of freedom (`DenDF`, denoted by ν_{2i}, Kenward and Roger, 1997) and a probablity value if these can be computed. They will be for the conditional Wald F statistic if it is reported. The `!DDF` i (see page 67) qualifier can be used to suppress the `DenDF` calculation (`!DDF -1`) or request a particular algorithmic method: `!DDF 1` for numerical derivatives, `!DDF 2` for algebraic derivatives. The value in the probability column (either `P_inc` or `P_con`) is computed from an $F_{\nu_{1i},\nu_{2i}}$ reference distribution. An approximation is used for computational convenience when calculating the `DenDF` for Conditional F statistics using numerical derivatives. The `DenDF` reported then relates to a maximal conditional incremental model (MCIM) which, depending on the model order, may not always coincide with the maximal conditional model (MCM) under which the conditional F statistic is calculated. The MCIM model omits terms fitted after any terms ignored for the conditional test (`I` after `.` in marginality pattern). In the example above, MCIM ignores `variety.sow` when calculating DenDF for the test of `water` and ignores `water.sow` when calculating DenDF for the test of `variety`. When `DenDF` is not available, it is often 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 F statistics is simplified when the adjusted variance matrix is used. Secondly, if measures of precision are required for $\hat{\boldsymbol{\tau}}$ or effects therein, those obtained from the adjusted variance matrix will generally be preferred. Unfortunately the Wald statistics are currently computed using an unadjusted variance matrix.

2.5.4 Approximate stratum variances

ASReml reports 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 computes approximate stratum degrees of freedom and stratum variances by a modified Cholesky diagonalisation of the average information matrix. That is, if $\boldsymbol{F}$ is the average information matrix for $\boldsymbol{\sigma}$, let $\boldsymbol{U}$ be an upper triangular matrix such that $\boldsymbol{F} = \boldsymbol{U}^{\mathsf{T}}\boldsymbol{U}$. We define

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

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

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

Thence

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

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

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

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

3 A guided tour

3.1 Introduction

This chapter presents a guided tour of ASReml, from data file preparation and basic aspects of the ASReml command file, to running an ASReml job and interpreting the output files. You are encouraged to read this chapter before moving to the later chapters;

- a real data example is used in this chapter for demonstration, see below,
- the same data are also used in later chapters,
- links to the formal discussion of topics are clearly signposted by margin notes.

This example is of a randomised block analysis of a field trial, and is only one of many forms of analysis that ASReml can perform. It is chosen because it allows an introduction to the main ideas involved in running ASReml. However some aspects of ASReml, in particular, pedigree files (see Chapter 9) and multivariate analysis (see Chapter 8) are only covered in later chapters.

ASReml is essentially a batch program with some optional interactive features. The typical sequence of operations when using ASReml is

- Prepare the data (typically using a spreadsheet or data base program)
- Export that data as an ASCII file (for example export it as a `.csv` (comma separated values) file from Excel)
- Prepare a job file with filename extension `.as`.
- Run the job file with ASReml
- Review the various output files
- revise the job and re run it, or

- extract pertinent results for your report.

You will need a file editor to create the command file and to view the various output files. On unix systems, `vi` and `emacs` are commonly used. Under Windows, there are several suitable program editors available such as ASReml-W and ConText mentioned in Section 1.3.

3.2 Nebraska Intrastate Nursery (NIN) field experiment

The yield data from an advanced Nebraska Intrastate Nursery (NIN) breeding trial conducted at Alliance in 1988/89 will be used for demonstration, see Stroup *et al.* (1994) for details. Four replicates of 19 released cultivars, 35 experimental wheat lines and 2 additional triticale lines were laid out in a 22 row by 11 column rectangular array of plots; the varieties were allocated to the plots using a randomised complete block (RCB) design. In field trials, complete replicates are typically allocated to consecutive groups of *whole* columns or rows. In this trial the replicates were not allocated to groups of whole columns, but rather, overlapped columns. Table 3.1 gives the allocation of varieties to plots in field plan order with replicates 1 and 3 in *ITALICS* and replicates 2 and 4 in **BOLD**.

Table 3.1: Trial layout and allocation of varieties to plots in the NIN field trial

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

3.3 The ASReml data file

The standard format of an ASReml data file is to have the data arranged in space, TAB or comma separated columns/fields with a line for each sampling unit. The columns contain covariates, factors, response variates (traits) and weight variables in any convenient order. This is the first 30 lines of the file `nin89.asd` containing the data for the NIN variety trial. The data are in field order (rows within columns) and an optional heading (first line of the file) has been included to document the file. In this case there are 11 space separated data fields (`variety`...`column`) and the complete file has 224 data lines, one for each variety in each replicate.

```
variety id pid raw repl nloc yield lat long row column
LANCER   1 1101 585 1 4 29.25 4.3 19.2 16 1
BRULE    2 1102 631 1 4 31.55 4.3 20.4 17 1
REDLAND 3 1103 701 1 4 35.05 4.3 21.6 18 1
CODY     4 1104 602 1 4 30.1 4.3 22.8 19 1
ARAPAHOE 5 1105 661 1 4 33.05 4.3 24 20 1
NE83404 6 1106 605 1 4 30.25 4.3 25.2 21 1
NE83406 7 1107 704 1 4 35.2 4.3 26.4 22 1
NE83407 8 1108 388 1 4 19.4 8.6 1.2 1 2
CENTURA 9 1109 487 1 4 24.35 8.6 2.4 2 2
SCOUT66 10 1110 511 1 4 25.55 8.6 3.6 3 2
COLT     11 1111 502 1 4 25.1 8.6 4.8 4 2
NE83498 12 1112 492 1 4 24.6 8.6 6 5 2
NE84557 13 1113 509 1 4 25.45 8.6 7.2 6 2
NE83432 14 1114 268 1 4 13.4 8.6 8.4 7 2
NE85556 15 1115 633 1 4 31.65 8.6 9.6 8 2
NE85623 16 1116 513 1 4 25.65 8.6 10.8 9 2
CENTURAK78 17 1117 632 1 4 31.6 8.6 12 10 2
NORKAN 18 1118 446 1 4 22.3 8.6 13.2 11 2
KS831374 19 1119 684 1 4 34.2 8.6 14.4 12 2
⋮
```

optional field labels
data for sampling unit 1
data for sampling unit 2

.
.
.

These data are analysed again in Chapter 7 using spatial methods of analysis, see model **3a** in Section 7.5. For spatial analysis using a separable error structure (see Chapter 2) the data file must first be augmented to specify the complete 22 row × 11 column array of plots. These are the first 20 lines of the augmented data file `nin89aug.asd` with 242 data rows. Note that ASReml 4 can automatically augment spatial data: see `!ROWFACTOR`, `!COLUMNFACTOR`.

```
variety id pid raw repl nloc yield lat long row column   optional field labels
LANCER 1 NA NA 1 4 NA 4.3 1.2 1 1                         file augmented by missing values
LANCER 1 NA NA 1 4 NA 4.3 2.4 2 1                         for first 15 plots and 3 buffer
LANCER 1 NA NA 1 4 NA 4.3 3.6 3 1                         plots and variety coded LANCER
LANCER 1 NA NA 1 4 NA 4.3 4.8 4 1                         to complete 22×11 array
LANCER 1 NA NA 1 4 NA 4.3 6 5 1                           .
LANCER 1 NA NA 1 4 NA 4.3 7.2 6 1                         .
LANCER 1 NA NA 1 4 NA 4.3 8.4 7 1                         .
LANCER 1 NA NA 1 4 NA 4.3 9.6 8 1
LANCER 1 NA NA 1 4 NA 4.3 10.8 9 1
LANCER 1 NA NA 1 4 NA 4.3 12 10 1
LANCER 1 NA NA 1 4 NA 4.3 13.2 11 1
LANCER 1 NA NA 1 4 NA 4.3 14.4 12 1
LANCER 1 NA NA 1 4 NA 4.3 15.6 13 1
LANCER 1 NA NA 1 4 NA 4.3 16.8 14 1                       buffer plots
LANCER 1 NA NA 1 4 NA 4.3 18 15 1                         between reps
LANCER 1 NA NA 2 4 NA 17.2 7.2 6 4
LANCER 1 NA NA 3 4 NA 25.8 22.8 19 6
LANCER 1 NA NA 4 4 NA 38.7 12.0 10 9
LANCER 1 1101 585 1 4 29.25 4.3 19.2 16 1                 original data
BRULE 2 1102 631 1 4 31.55 4.3 20.4 17 1                  .
REDLAND 3 1103 701 1 4 35.05 4.3 21.6 18 1                .
CODY 4 1104 602 1 4 30.1 4.3 22.8 19 1                    .
⋮
```

Note that

- the `pid`, `raw`, `repl` and `yield` data for the missing plots have all been made `NA` (one of the three missing value indicators in ASReml, see Section 4.2),
- `variety` is coded `LANCER` for all missing plots; one of the variety names must be used but the particular choice is arbitrary.

3.4 The ASReml command file

By convention an ASReml command file has a `.as` extension. The file defines

- a title line to describe the job,
- labels for the data fields in the data file and the name of the data file,
- the linear mixed model and the variance model(s) if required,
- output options including directives for tabulation and prediction.

Below is the ASReml command file for an RCB analysis of the NIN field trial data highlighting the main sections. Note the order of the main sections.

```
                          title line ⟶   NIN Alliance trial 1989
               data field definition⟶     variety !A
                                    .     id
                                    .     pid
                                    .     raw
                                          repl 4
                                          nloc
                                          yield
                                          lat
                                          long
                                          row 22
               data field definition ⟶    column 11
      data file name and qualifiers⟶    nin89.asd !skip 1
                tabulate statement⟶    tabulate yield ~ variety
        linear mixed model definition⟶    yield ~ mu variety !r idv(repl)
residual variance model specification⟶    residual idv(units)
                  predict statement⟶    predict variety
```

3.4.1 Generating a template

ASReml can generate a basic command file, a template for you to modify, from the data file if the data file has suitable field (variable) names in the first line. The requirements are

- the data file have file name extension `asd`, `csv`, `dat` or `txt`;
- there is not a matching command file already existing;
- the first line of the file contains a ’name’ for each field;
- the ’name’ must begin with a letter; it may contain numbers and the underscore character but not any of the characters `+-.,:;$#\*/^!|&’"<>=~{}[]();`
- the ’name’ may be terminated with `!P` to indicate a Pedigree factor, `!A` to indicate a

alphanumerically coded factor, `!I` to indicated a factor where the numbers are to be treated as labels for the levels, and `!` where the numbers are the actual levels.

- If none of the 'names' are indicated as factors using the `!` mechanism, ASReml will scan the first few lines of data and try and identify alphanumeric, integer and simple factors.

Always check the template as it is likely some variates have been misclassified as factors.

The template file created by running ASReml on the `nin89.asd` file looks like

```
# !WORKSPACE 100 !RENAME !ARGS // !DOPART $1
Title: nin89.
#variety,id,pid,raw,rep,nloc,yield,lat,long,row,column
#LANCER,1,1101,585,1,4,29.25,4.3,19.2,16,1
#BRULE,2,1102,631,1,4,31.55,4.3,20.4,17,1
#REDLAND,3,1103,701,1,4,35.05,4.3,21.6,18,1
#CODY,4,1104,602,1,4,30.1,4.3,22.8,19,1
 variety  !A      # CODY
 id  *       # 4
 pid  !I     # 1104
 raw  !I     # 602
 rep  *      # 1
 nloc  *      # 4
 yield       # 30.1
 lat       # 4.3
 long       # 22.8
 row  !I     # 19
 column  *       # 1
# Check/Correct these field definitions.
nin89.asd  !SKIP 1
yield  ~ mu ,         # Specify fixed model
      !r          # Specify random model
residual units
```

We need to change the `!I` associated with `row` to `*` because the row numbers are actually positions, not just labels which could be taken in any order. Note that ASReml displays a data value beside each name to make it easier to confirm the labelling.

3.4.2 The title line

The first text (non-blank, non control) line in an ASReml command file is taken as the title for the job and is purely descriptive for future reference.

```
NIN Alliance trial 1989
 variety !A
 id ⋮
```

3.4.3 Reading the data

The data fields are defined before the data file name is specified. Field definitions must be given for all fields in the data file and in the order in which they appear in the data file. Note that, in previous releases data field definitions had to be indented but in Release 4 this condition has been relaxed and is not required. In this case there are 11 data fields (`variety` ... `column`) in `nin89.asd`, see Section 3.3.

```
NIN Alliance trial 1989
 variety !A
 id
 pid
 raw
 repl 4
 nloc
 yield
 lat
 long
 row 22
 column 11
nin89.asd !skip 1 ⋮
```

The `!A` after `variety` tells ASReml that the first field is an alphanumeric factor and the 4 after `repl` tells ASReml that the field called `repl` (the fifth field read) is a numeric factor with 4 levels coded 1:4. Similarly for `row` and `column`. The other fields include variates (`yield`) and various other variables.

3.4.4 The data file line

The data file name is specified immediately after the last data field definition. Data file qualifiers that relate to data input and output are also placed on this line if they are required. In this example, `!skip 1` tells ASReml to ignore (skip) the first line of the data file `nin89.asd`, the line containing the field labels.

The data file line can contain qualifiers that control other aspects of the analysis. These qualifiers are presented in Section 5.8.

```
NIN Alliance trial 1989  variety !A
 id
 pid
⋮
 row 22
 column 11
nin89.asd !skip 1
tabulate yield ~ variety
yield ~ mu variety !r idv(repl)
residual idv(units)
predict variety
```

3.4.5 Tabulation

The `tabulate` statements are optional.They provide a simple way of exploring the structure of a data. They should appear immediately before the model line. In this case the 56 simple variety means for yield are formed and written to a `.tab` output file. See Chapter 10 for a discussion of tabulation.

```
⋮
 column 11
nin89.asd !skip 1
tabulate yield ~ variety
yield ~ mu variety !r idv(repl)
residual idv(units)
predict variety
```

3.4.6 Specifying the terms in the mixed model

The linear mixed model is specified as a list of model terms and qualifiers. All elements must be space separated. ASReml accommodates a wide range of analyses. See Section 2.1 for a brief discussion and general algebraic formulation of the linear mixed model. The model specified here for the NIN data is a simple random effects RCB model having *fixed* variety effects and *random* replicate effects. The reserved word `mu` fits a constant term (intercept), `variety` fits a fixed variety effect and `repl` fits a random replicate effect because the `!r` qualifier tells ASReml to fit the terms that follow as random effects.

```
NIN Alliance trial 1989  variety !A
⋮
 column 11
nin89.asd !skip 1
tabulate yield ~ variety
yield ~ mu variety !r idv(repl)
residual idv(units)
predict variety
```

3.4.7 Variance structures

There are two variance structures to be specified and two variance components to be estimated. The first structure is for the replicate (`repl`) effects. These effects are IID distributed and `idv(repl)` denotes this and estimates one variance component associated with these effects. The other is associated with the residual effects, which are again assumed to be IID distributed. This is formally specified here by the line `residual idv(units)` where `residual` is the name of the directive that specifies the variance structure for the residuals, and `units` is the reserved word specifying a factor with a level for every experimental unit. The default variance structure is always uncorrelated effects with a common variance and so `idv(repl)` and `idv(units)` can be reduced to simply `repl` and `units`. See Chapter 7 for a lengthy discussion on variance modelling in ASReml.

```
NIN Alliance trial 1989  variety !A
⋮
 column 11
nin89.asd !skip 1
tabulate yield ~ variety
yield ~ mu variety !r idv(repl)
residual idv(units)
predict variety
```

3.4.8 Prediction

Predict statements appear after the model statement. In this case the 56 variety means for yield as predicted from the fitted model would be formed and returned in the `.pvs` output file. See Chapter 10 for a detailed discussion of prediction in ASReml.

```
NIN Alliance trial 1989  variety !A
:
 column 11
nin89.asd !skip 1
tabulate yield ~ variety
yield ~ mu variety !r idv(repl)
residual idv(units)
predict variety
```

3.5 Running the job

Assuming you have located the `nin89.asd` file (under Windows it will typically be located in *ASRemlPath*/`Examples`; we suggest copying the data file to the users workspace as the `Examples` folder is sometimes write protected) and created the `ASCII` command file `nin89.as` as described in the previous section and in the same folder, you can run the job. *ASRemlPath* is typically `C:\Program Files\ASReml4` under Windows. Installation details vary with the implementation and are distributed with the program. You could use ASReml-W or ConText to create `nin89.as`. These programs can then run ASReml directly after they have been configured for ASReml. An ASReml job is also run from a command line or by 'clicking' the `.as` file in Windows Explorer.

The basic command to run an ASReml job is

ASRemlPath`/bin/ASReml` *basename*[`.as`]

where *basename*[`.as`] is the name of the command file. Typically, a system PATH is defined which includes *ASRemlPath*`/bin/` so that just the program name `ASReml` is required at the command prompt. For example, the command to run `nin89.as` from the command prompt when attached to the appropriate folder is

```
ASReml nin89.as
```

However, if the path to ASReml is not specified in your system's PATH environment variable, the path must also be given, and the path is required when configuring ASReml-W or Context.

In this guide we assume the command file has a filename extension `.as`. ASReml also recognises the filename extension `.asc` as an ASReml command file. When these are used, the extension (`.as` or `.asc`) may be omitted from *basename*`.as` in the command line if there is no file in the working directory with the name *basename*. The *options* and *arguments* that can be supplied on the command line to modify a job at run time are described in Chapter 11.

3.6 Description of output files

A series of output files are produced with each ASReml run. Nearly all files, all that contain user information, are ASCII files and can be viewed in any ASCII editor including ConText, ASReml-W and NotePad. The primary output from the nin89.as job is written to nin89.asr. This file contains a summary of the data, the iteration sequence, estimates of the variance parameters and an a table of Wald F statistics for testing fixed effects. The estimates of all the fixed and random effects are written to nin89.sln. The residuals, predicted values of the observations and the diagonal elements of the hat matrix (see Chapter 2) are returned in nin89.yht, see Section 14.3. Other key files produced by this job include the .aov, .pvs, .res, .tab, .sln and .yht files, see Section 14.4.

3.6.1 The .asr file

Below is nin89.asr with pointers to the main sections. The first line gives the version of ASReml used (in square brackets) and the title of the job. The second line gives the build date for the program and indicates whether it is a 32bit or 64bit version. The third line gives the date and time that the job was run and reports the size of the workspace. The general announcements box (outlined in asterisks) at the top of the file notifies the user of current release features. The remaining lines report a data summary, the iteration sequence, the estimated variance parameters and a table of Wald F statistics. The final line gives the date and time that the job was completed and a statement about convergence.

```
ASReml 3.1 [01 Jan 2011]   NIN alliance trial 1989\\ job heading
  Build cm [25 Oct 2011]   64 bit
04 Nov 2011 21:14:28.404     32 Mbyte Linux (x64)  nin89
Licensed to: Cargo Vale Olives/Univ of Wollongong    31-jul-2012
***********************************************************
* Contact support@asreml.co.uk for licensing and support   *
***************************************************** ARG *
Folder: /home/gilmoua/W7drive/Users/Public/ASReml/asr3/ug3/Manex4
variety !A
QUALIFIERS: !SKIP 1
Reading nin89.asd  FREE FORMAT skipping     1 lines

Univariate analysis of yield
Summary of 224 records retained of 224 read data summary

 Model term          Size #miss #zero   MinNon0    Mean      MaxNon0  StndDevn
  1 variety            56     0     0      1    28.5000         56
  2 id                        0     0  1.000      28.50      56.00      16.20
  3 pid                       0     0  1101.      2628.      4156.      1121.
  4 raw                       0     0  21.00      510.5      840.0      149.0
  5 repl                4     0     0      1     2.5000          4
  6 nloc                      0     0  4.000      4.000      4.000      0.000
  7 yield          Variate    0     0  1.050      25.53      42.00      7.450
  8 lat                        0     0  4.300      27.22      47.30      12.90
  9 long                       0     0  1.200      14.08      26.40      7.698
 10 row                22      0     0      1    11.7321        22
 11 column             11      0     0      1     6.3304        11
 12 mu                          1
Forming      61 equations:  57 dense.
Initial updates will be shrunk by factor    0.400
Notice:      1 singularities detected in design matrix.
  1 LogL=-454.807     S2=  50.329        168 df    0.1000
  2 LogL=-454.635     S2=  50.073        168 df    0.1219
  3 LogL=-454.513     S2=  49.818        168 df    0.1537
  4 LogL=-454.471     S2=  49.622        168 df    0.1899
  5 LogL=-454.469     S2=  49.584        168 df    0.1989
  6 LogL=-454.469     S2=  49.582        168 df    0.1993
Final parameter values                        0.1993

         - - - Results from analysis of yield - - -
Akaike Information Criterion     912.94 (assuming 2 parameters).
Bayesian Information Criterion    919.19

         Approximate stratum variance decomposition
Stratum     Degrees-Freedom   Variance      Component Coefficients
idv(repl)              3.00    603.100        56.0     1.0
Residual Variance    165.00    49.5824         0.0     1.0

Model_Term                             Gamma         Sigma   Sigma/SE   % C
idv(repl)               IDV_V    4  0.199323       9.88291    % 1.12    0 P parameter
idv(units)                    224 effects                                   estimates
Residual                SCA_V  224  1.000000       49.5824       9.08    0 P

                                  Wald F statistics
    Source of Variation           NumDF     DenDF    F-inc            P-inc testing
 12 mu                                1       3.0   242.05            <.001 fixed
  1 variety                          55     165.0     0.88            0.712 effects
Notice: The DenDF values are calculated ignoring fixed/boundary/singular
            variance parameters using algebraic derivatives.
  5 repl                                  4 effects fitted
Finished: 04 Nov 2011 21:14:29.242   LogL Converged
```

3.6.2 The `.sln` file

The following is an extract from `nin89.sln` containing the estimated variety effects, intercept and random replicate effects in this order (column 3) with standard errors (column 4). Note that the variety effects are returned in the order of their first appearance in the data file, see replicate 1 in Table 3.1.

```
Model_Term            Level                 Effect    seEffect
 variety              LANCER                 0.000       0.000
 variety              BRULE                 -2.487       4.979
 variety              REDLAND                1.938       4.979
 variety              CODY                  -7.350       4.979
 variety              ARAPAHOE              0.8750       4.979
 variety              NE83404               -1.175       4.979
 variety              NE83406               -4.287       4.979
 variety              NE83407               -5.875       4.979
 variety              CENTURA               -6.912       4.979
 variety              SCOUT66               -1.037       4.979
 variety              COLT                  -1.562       4.979
 variety              NE83498                1.563       4.979
 variety              NE84557               -8.037       4.979
 variety              NE83432               -8.837       4.979
⋮
 variety              NE87615               -2.875       4.979
 variety              NE87619                2.700       4.979
 variety              NE87627               -5.337       4.979
 mu                             1            28.56       3.856
 repl                           1            1.880       1.755
 repl                           2            2.843       1.755
 repl                           3          -0.8713       1.755
 repl                           4           -3.852       1.755
```

3.6.3 The `.yht` file

The following is an extract from `nin89.yht` containing the predicted values of the observations (column 2), the residuals (column 3) and the diagonal elements of the hat matrix. This final column can be used in tests involving the residuals, see Section 2.4 under Diagnostics.

```
  Record          Yhat      Residual           Hat
       1        30.442        -1.192         13.01
       2        27.955         3.595         13.01
       3        32.380         2.670         13.01
       4        23.092         7.008         13.01
       5        31.317         1.733         13.01
       6        29.267        0.9829         13.01
       7        26.155         9.045         13.01
       8        24.567        -5.167         13.01
       9        23.530        0.8204         13.01
⋮
     222        16.673         9.877         13.01
     223        24.548         1.052         13.01
     224        23.786         3.114         13.01
```

3.7 Tabulation, predicted values and functions of the variance components

It may take several runs of ASReml to determine an appropriate model for the data, that is, the fixed and random effects that are important. During this process you may wish to explore the data by simple tabulation. Having identified an appropriate model, you may then wish to form predicted values or functions of the variance components. The facilities in ASReml to form predicted values and functions of the variance components are described in Chapters 10 and 13 respectively. Our example only includes tabulation and prediction.

The statement

```
tabulate yield ~ variety
```

in `nin89.as` results in `nin89.tab` as follows:

```
 NIN alliance trial 1989                                        11 Jul 2005 13:55:21

                     Simple tabulation of yield


  variety
  LANCER                     28.56
  BRULE                      26.07
  REDLAND                    30.50
  CODY                       21.21
  ARAPAHOE                   29.44
  NE83404                    27.39
  NE83406                    24.28
  NE83407                 22.69
  CENTURA                 21.65
  SCOUT66                 27.52
  COLT                    27.00
 ⋮
  NE87522                 25.00
  NE87612                 21.80
  NE87613                 29.40
  NE87615                 25.69
  NE87619                 31.26
  NE87627                 23.23
```

The

```
predict variety
```

statement after the model statement in `nin89.as` results in the `nin89.pvs` file displayed below (some output omitted) containing the 56 predicted variety means, also in the order in which they first appear in the data file (column 2), together with standard errors (column 3). An average standard error of difference among the predicted variety means is displayed immediately after the list of predicted values. As in the `.asr` file, date, time and trial information are given the title line. The `Ecode` for each prediction (column 4) is usually `E` indicating the prediction is of an estimable function. Predictions of non-estimable functions are usually not printed, see Chapter 10.

```
NIN alliance trial 1989                                     04 Apr 2008 17:00:47
                                                            nin89

Ecode is E for Estimable, * for Not Estimable

---- ---- ---- ---- ---- ---- ---- ----   1 ---- ---- ---- ---- ---- ---- ----
Predicted values of yield
The predictions are obtained by averaging across the hypertable
       calculated from model terms constructed solely from factors
       in the averaging and classify sets.
The ignored set: repl
Use !AVERAGE to move table factors into the averaging set.

 variety              Predicted_Value Standard_Error Ecode
 LANCER                       28.5625         3.8557 E
 BRULE                        26.0750         3.8557 E
 REDLAND                      30.5000         3.8557 E
 CODY                         21.2125         3.8557 E
 ARAPAHOE                     29.4375         3.8557 E
 NE83404                      27.3875         3.8557 E
 NE83406                      24.2750         3.8557 E
 NE83407                      22.6875         3.8557 E
```

```
  CENTURA                     21.6500         3.8557 E
  SCOUT66                     27.5250         3.8557 E
  COLT                        27.0000         3.8557 E
⋮
  NE87613                     29.4000         3.8557 E
  NE87615                     25.6875         3.8557 E
  NE87619                     31.2625         3.8557 E
  NE87627                     23.2250         3.8557 E
  SED: Overall Standard Error of Difference   4.979
```

4 Data file preparation

4.1 Introduction

The first step in an ASReml analysis is to prepare the data file. Data file preparation is discussed in this chapter using the NIN example of Chapter 3 for demonstration. The first 25 lines of the data file are as follows:

```
variety id pid raw repl nloc yield lat long row column
BRULE 2 1102 631 1 4 31.55 4.3 20.4 17 1
REDLAND 3 1103 701 1 4 35.05 4.3 21.6 18 1
CODY 4 1104 602 1 4 30.1 4.3 22.8 19 1
ARAPAHOE 5 1105 661 1 4 33.05 4.3 24 20 1
NE83404 6 1106 605 1 4 30.25 4.3 25.2 21 1
NE83406 7 1107 704 1 4 35.2 4.3 26.4 22 1
NE83407 8 1108 388 1 4 19.4 8.6 1.2 1 2
CENTURA 9 1109 487 1 4 24.35 8.6 2.4 2 2
SCOUT66 10 1110 511 1 4 25.55 8.6 3.6 3 2
COLT 11 1111 502 1 4 25.1 8.6 4.8 4 2
NE83498 12 1112 492 1 4 24.6 8.6 6 5 2
NE84557 13 1113 509 1 4 25.45 8.6 7.2 6 2
NE83432 14 1114 268 1 4 13.4 8.6 8.4 7 2
NE85556 15 1115 633 1 4 31.65 8.6 9.6 8 2
NE85623 16 1116 513 1 4 25.65 8.6 10.8 9 2
CENTURK78 17 1117 632 1 4 31.6 8.6 12 10 2
NORKAN 18 1118 446 1 4 22.3 8.6 13.2 11 2
KS831374 19 1119 684 1 4 34.2 8.6 14.4 12 2
TAM200 20 1120 422 1 4 21.1 8.6 15.6 13 2
NE86482 21 1121 560 1 4 28 8.6 16.8 14 2
HOMESTEAD 22 1122 566 1 4 28.3 8.6 18 15 2
LANCOTA 23 1123 514 1 4 25.7 8.6 19.2 16 2
NE86501 24 1124 635 1 4 31.75 8.6 20.4 17 2
NE86503 25 1125 840 1 4 42 8.6 21.6 18 2
⋮
```

4.2 The data file

The standard format of an ASReml data file is to have the data arranged in columns/fields with a single line for each sampling unit. The columns contain variates and covariates (numeric), factors (alphanumeric), traits (response variables) and weight variables in any order that is convenient to the user. The data file may be free format, fixed format or a binary file.

4.2.1 Free format data files

The data are read free format (SPACE, COMMA or TAB separated) unless the file name has extension `.bin` for real binary, or `.dbl` for double precision binary (see below). Important points to note are as follows:

- files prepared in EXCEL must be saved to comma or tab-delimited form.
- blank lines are ignored,
- column headings, field labels or comments may be present at the top of the file (See **Generating a template** on page 29) provided that the `!skip` qualifier (Table 5.2) is used to skip over them,
- `NA`, * and `.` are treated as coding for *missing values* in free format data files;
  - if missing values are coded with a unique data *value* (for example, 0 or -9), use the transformation `!M` *value* to flag them as *missing* or `!DV` *value* to drop the data record containing them (see Table 5.1),
- comma delimited files whose file name ends in `.csv` or for which the `!CSV` qualifier is set recognise empty fields as missing values,
  - a line beginning with a comma implies a preceding missing value,
  - consecutive commas imply a missing value,
  - a line ending with a comma implies a trailing missing value,
  - if the filename does not end in `.csv` and the `!CSV` qualifier is not set, commas are treated as white space,
- TAB delimited files recognise empty fields as missing values
- characters following `#` on a line are ignored so this character may not be used except to flag trailing comments on the ends of lines, or to comment out data records, unless `!SPECIALCHAR` is specified, see Section 5.4.2,
- adjacent lines can be concatenated and written on one line using `//`. For example,

```
line_1
line_2
⋮
line_n
```

can be written on one line as

`line_1 // line_2 // ... line_n`

This can aid legibility of the input file. Note that everything, including `//`, after the first `#` on a line is intepreted as a comment,

- blank spaces, tabs and commas must not be used (embedded) in alphanumeric fields unless the label is enclosed in quotes, for example, the name `Willow Creek` would need to be appear in the data file as `‘Willow Creek‘` to avoid an error,

- the `$` symbol must not be used in the data file,

- alphanumeric factor level labels have a default size of 16 characters. Use the `!LL` *size* qualifier to extend the size of factor labels stored.

- extra data fields on a line are ignored,

- if there are fewer data items on a line than ASReml expects, the remainder are taken from the following line(s) except in `.csv` files were they are taken as missing. If you end up with half the number of records you expected, this is probably the reason,

- all lines beginning with `!` followed by a blank are copied to the `.asr` file as comments for the output; their contents are ignored,

4.2.2 Fixed format data files

The format must be supplied with the `!FORMAT` qualifier which is described in Table 5.5. However, if all fields are present and are separated, the file can be read free format.

4.2.3 Preparing data files in Excel

Many users find it convenient to prepare their data in Excel, Access or some other database. Such data must be exported from these programs into either `.csv` (Comma separated values) or `.txt` (TAB separated values) form for ASReml to read it. ASReml can convert an `.xls` file to a `.csv` file. When ASReml is invoked with an `.xls` file as the filename argument and there is no `.csv` file or `.as` with the same basename, it exports the first sheet as a `.csv` file and then generates a template `.as` command file from any column headings it finds (see page 195). It will also convert a Genstat `.gsh` spreadsheet file to `.csv` format. The data extracted from the `.xls` file are labels, numerical values and the results from formulae. Empty rows at the start and end of a block are trimmed, but empty rows in the middle of a block are kept. Empty columns are ignored. A single row of labels as the first non-empty row in the block will be taken as column names. Empty cells in this row will have default names `C1`, `C2` etc. assigned. Missing values are commonly represented in ASReml data files by `NA`, `*` or `.`. ASReml will also recognise empty fields as missing values in `.csv` (`.xls`) files.

4.2.4 Binary format data files

Conventions for binary files are as follows:

- binary files are read as unformatted Fortran binary in single precision if the filename has a `.bin` or `.BIN` extension,

- Fortran binary data files are read in double precision if the filename has a `.dbl` or `.DBL` extension,

- ASReml recognises the value `-1e37` as a missing value in binary files,

- Fortran binary in the above means all real (`.bin`) or all double precision (`.dbl`) variables; mixed types, that is, integer and alphabetic binary representation of variables is not allowed in binary files,

- binary files can only be used in conjunction with a pedigree file if the pedigree fields are coded in the binary file so that they correspond with the pedigree file (this can be done using the `!SAVE` option in ASReml to form the binary file, see Table 5.5), or the identifiers are whole numbers less than 9,999,999 and the `!RECODE` qualifier is specified (see Table 5.5).

5 Command file: Reading the data

5.1 Introduction

In the code box to the right is the ASReml command file `nin89a.as` for a spatial analysis of the Nebraska Intrastate Nursery (NIN) field experiment introduced Chapter 3. The lines that are highlighted in bold/blue type relate to reading in the data. In this chapter we use this example to discuss reading in the data in detail.

```
NIN Alliance Trial 1989
 variety !A # Alphanumeric
 id
 pid
 raw
 repl 4
 nloc
 yield
 lat
 long
 row 22
 column 11
nin89aug.asd !skip 1
yield ~ mu variety
residual idv(units)
```

Notice the in-line comment indicated by the `#`.

5.2 Important rules

In the ASReml command file

- all blank lines are ignored,
- `#` is used to annotate the input; all characters following a `#` symbol on a line are ignored,
- lines beginning with `!` followed by a blank are copied to the `.asr` file as comments for the output,
- a blank is the usual separator; TAB is also a separator,

5.2 Important rules

- a comma as the last character on the line is sometimes used to indicate that the current list is continued on the next line; a comma is not needed when ASReml knows how many values to read,

- reserved words used in specifying the linear model (Table 6.1) are case sensitive; they need to be typed exactly as defined: they may not be abbreviated.

- a qualifier is a letter sequence preceded by `!` which sets an option;
  - some qualifiers require arguments,
  - qualifiers must appear on the correct context,
  - qualifier identifiers are not case sensitive,
  - most qualifier identifiers may be truncated to 3 characters.

5.3 Title line

The first 40 characters of the first nonblank text line in an ASReml command file are taken as a title for the job. Use this to document the analysis for future reference. An optional qualifier line (see section 11.3) may precede the title line. It is recognised by the presence of the qualifier prefix letter `!`. Therefore the title MUST NOT include an exclamation mark.

```
NIN Alliance Trial 1989
 variety !A
 id
 pid
⋮
```

5.4 Specifying and reading the data

Typically, a data record consists of all the information pertaining to an experimental unit (plot, animal, assessment). Data field definitions manage the process of converting the fields as they appear in the data file to the internal form needed by ASReml. This involves mapping (coding) factors, general transformations, skipping fields and discarding unnecessary records. If the necessary information is not in a single file, the `MERGE` facility (see Chapter 12) may help.

Variables are defined immediately after the job title. These definitions indicate how each field in the data file is handled as it is read into ASReml. Transformations can be used to create additional variables. Users can explicity nominate how many are read with the `!READ` qualifier described in Table 5.5. No more than 10,000 variables may be read or formed.

Data field definitions

- should be given for all fields in the data file; fields can be skipped and fields (on the end of a data line) without a field definition are ignored; if there are not enough data fields on a data line, the remainder are taken from the next line(s),
- must be presented in the order they appear in the data file,
- can appear with other definitions on the same line,
- data fields can be transformed (see below):
- additional variables can be created by transformation qualifiers.

```
NIN Alliance Trial 1989
 variety !A
 id
 pid
 raw
 repl 4
 nloc
 yield
 lat
 long
 row 22
 column 11
nin89aug.asd !skip 1
yield ∼ mu variety
⋮
```

5.4.1 Data field definition syntax

Data field definitions appear in the ASReml command file in the form

SPACE *label* [*field_type*] [*transformations*]

- SPACE – is now optional

- *label*
  - is an alphanumeric string to identify the field,

  - has a maximum of 31 characters although only 20 are ever printed/displayed,

  - must begin with a letter,

  - must not contain the special characters ., *, :, /, !, #, $, | or (,

  - reserved words (Table 6.1 and Table 7.6) must not be used,

  - `!CSKIP [`*c*`]` can be used to skip *c* (default 1) data fields.

- *field_type* defines how a variable is interpreted as it is read and whether it is regarded as a factor or variate if specified in the linear model,
  - for a variate, leave *field_type* blank or specify `1`,

  - for a model factor, various qualifiers are required depending on the form of the factor coding where *n* is the number of levels of the factor and *s* is a list of labels (or the name of a file containing the labels one per row) to be assigned to the levels:

    `*` or *n* is used when the data field has values `1`... *n* directly coding for the factor unless the levels are to be labelled (see `!L`),
    `Row * # 1:12` for example

    `!L` *s* is used when the data field is numeric with values *1... n* and labels are to be assigned to the *n* levels, for example
    `Sex !L Male Female`

    `!A` [*n*] is required if the data field is alphanumeric, for example
    `Location !A # names`
    Specify *n* if there are more than 1000 classes over all class/factor variables indicating the expected number for this factor.

`!A !L` *s* is used if the data field is alphanumeric and must be coded in a particular order to set the order of the levels. For example `SNP !A !L C:C C:T T:T` defines the levels over-riding the default, data dependent order. If there are many labels, they may be written over several lines by using a trailing comma to indicate continuation of the list. New R4 Alternatively, the labels may be listed in a file. If the filename includes embedded blanks, or has no file extension, it must be enclosed in quotes:

```
Genotype !A !L MyNames.txt
Genotype !A !L 'My Names.txt'
Genotype !A !L 'MyNames'
```

Use a `!SKIP` qualifier after the filename to skip any heading lines. Names found in the data that are not included are simply appended to the list of levels as they are discovered by ASReml. An example of this would be for a genotype factor with 6 levels appearing in the data file in the order `genb6 gena1 gena5 genb2 genb4 gena3`. In this case

```
Genotype !A !L gena1 genb2 gena3 genb4
```

would result in the levels of `Genotype` being ordered `gena1 genb2 gena3 genb4 genb6 gena5`.

`!I` $[n]$ is required if the data is numeric defining a factor but not $1 \ldots n$; `!I` must be followed by n if more than 1000 codes are present,

```
Year !I # 1995 1996
```

`!AS` p is required if the data field has level names in common with a previous `!A` or `!I` factor p and is to be coded identically, for example in a plant diallel experiment

```
Male !A 22 Female !AS Male # integrated coding
```

`!P` indicates the special case of a pedigree factor; ASReml will determine whether the identifiers are integer or alphanumeric from the pedigree file qualifiers, and set the levels after reading the pedigree file, see Section 9.3,

```
Animal !P # coded according to pedigree file
```

A warning is printed if the nominated value for n does not agree with the actual number of levels found in the data; if the nominated value is too small the correct value is used.

- for a group of m variates or factor variables

!G *m* [*l*] is used when *m* contiguous data fields comprise a set to be used together. The variables will be treated as factor variables if the second argument (*l*) setting the number of levels is present (it may be *). For example

```
 ⋮                     is equivalent to     ⋮
 X1 X2 X3 X4 X5 y                           X !G 5 y
data.dat                                   data.dat
y ∼ mu X1 X2 X3 X4 X5                      y ∼ mu X
```

- !DATE specifies the field has one of the date formats *dd/mm/yy*, *dd/mm/ccyy*, *dd-Mon-yy*, *dd-Mon-ccyy* and is to be converted into a Julian day where *dd* is a 1 or 2 digit day of the month, *mm* is a 1 or 2 digit month of the year, *Mon* is a three letter month name (Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec), *yy* is the year within the century (00 to 99), *cc* is the century (19 or 20). The separators '/' and '-' must be present as indicated. The dates are converted to days starting 1 Jan 1900. When the century is not specified, *yy* of 0-32 is taken as 2000-2032, 33-99 taken as 1933-1999.

- !DMY specifies the field has one of the date formats *dd/mm/yy* or *dd/mm/ccyy* and is to be converted into a Julian day.

- !MDY specifies the field has one of the date formats *mm/dd/yy* or *mm/dd/ccyy* and is to be converted into a Julian day.

- !TIME specifies the field has the time format *hh:mm:ss.* and is to be converted to seconds past midnight where *hh* is hours (0 to 23), *mm* is minutes (0-59) and *ss* is seconds (0 to 59). The separator ':' must be present.

- *transformations* are described below.

5.4.2 Storage of alphabetic factor labels

Space is allocated dynamically for the storage of alphabetic factor labels with a default allocation being 2000 labels of 16 characters long. If there are large !A factors (so that the total across all factors will exceed 2000), you must specify the anticipated size (within say 5%) of the larger factors.

If some labels are longer then 16 characters and the extra characters are significant, you must lengthen the space for each label by specifying !LL *c* e.g.

```
cross !A 2300 !LL 48
```

indicates the factor cross has about 2300 levels and needs 48 characters to hold the level names; only the first 20 characters of the names are ever printed.

!PRUNE on a field definition line means that if fewer levels are actually present in the factor than were declared, ASReml will reduce the factor size to the actual number of levels. Use !PRUNEALL for this action to be taken on the current and subsequent factors up to (but not including) a factor with the `!PRUNEOFF` qualifier. The user may overestimate the size for large ALPHA and INTEGER coded factors so that ASReml reserves enough space for the list. Using `!PRUNE` will mean the extra (undefined) levels will not appear in the `.sln` file. Since it is sometimes necessary that factors not be pruned in this way, for example in pedigree/GIV factors, pruning is only done if requested.

Normally a # character in the data file will have the effect of eliminating whatever text follows on the line. This means that ordinarily the # character may not be included in the name of the level of an alphanumeric variable. The qualifier `!SPECIALCHAR` cancels the normal meaning of the # character in an input file so that it can be included in the name of a level of an alphanumeric or pedigree variable. If class names are being predefined, the qualifier `!SPECIALCHAR` must appear before the class names are read in.

5.4.3 Ordering factor levels

The default order for factor levels when factors are declared with `!I` and `!A` is the order the levels are encountered in the data file. `!SORT` declared after `!A` or `!I` on a field definition line will cause ASReml to fit the levels in (numeric) alphabetical order although they are defined in some other order. To control the order levels are defined, the level names must be prespecified using the `!L` *s* qualifier (applies only to factors declared `!A`). Thus for a variable `SEX` coded as `Male` and `Female`, declared `SEX !A` , the user cannot know whether it will be coded `1=Male, 2=Female` or `1=Female, 2=Male` without looking to see which occurs first in the data file. However declaring it as `SEX !A !L Male Female` will mean `Male` is coded 1; `Female` is coded 2. If it is declared as `SEX !A !SORT` , the coding order is unspecified but ASReml creates a lookup table after reading the data to arrange levels in sorted order and uses this sorted order when forming the design matrices. Consequentially, with the `!SORT` qualifier, the order of fitted effects will be `1=Female, 2=Male` in the analysis regardless of which appears first in the file.

It will generally be preferable to presepecify the levels than to use `!SORT` because most other references to particular levels of factors will refer to the unsorted levels. Therefore users should verify that ASReml has made the correct interpretation when nominating specific levels of `!SORT`ed factors. In particular any transformations are performed as the data is read in and before the sorting occurs.

`!SORTALL` means that the levels of this and subsequent factors are to be sorted.

5.4.4 Skipping input fields

This is particularly useful in large files with alphabetic fields that are not needed as it saves ASReml the time required to classify the alphabetic labels. New R4 `!CSKIP` *f* can be used to skip *f* fields. Thus

```
!CSKIP 1 A B
```

skips the first data field and reads the second and third fields into variables A and B, and

```
!CSKIP Sire !I !CSKIP 2 Y
```

will define two variables, Sire taken from the second data field and Y taken from the fifth data field. Also, !SKIP *f* will skip *f* data fields BEFORE reading this field. Thus

```
Sire !I !SKIP 1 Y !SKIP 2
```

achieves the same result but in a less obvious way! These qualifiers are ignored when reading binary data.

Important Using the !SKIP qualifier in association with the specification of a file to be read in allows initial lines of the file to be skipped. !SKIP can also be used to skip columns when reading in a data file. Use of !CSKIP for skipping data fields is recommended to avoid confusion.

5.5 Transforming the data

Transformation is the process of modifying the data (for example, dividing all of the data values in a field by 10), forming new variables (for example, summing the data in two fields) or creating temporary data (for example, a test variable used to discard some records from analysis and subsequently discarded). Occasional users may find it easier to use a spreadsheet to calculate derived variables than to modify variables using ASReml transformations.

Transformation qualifiers are listed after data field labels (and the *field_type* if present). They define an operation (e.g. +), often involving an argument (a constant or another variable), which is performed on a *target* variable. By default the *target* is the current field, but can be changed with the !TARGET qualifier. For a !G group of variables, the target is the first variable in the set.

Using transformations will be easier if you understand the process. As ASReml parses the variable definitions, it sequentially assigns them column positions in the internal data vector. It notes which is the last variable which is not created by (say the !=) transformation, and that determines how many fields are read from the data file (unless overridden by !READ qualifier in Table 5.2). ASReml actually reads the data file *after* parsing the model line. It reads a line into a temporary vector, performs the transformations in that vector, and saves the values that relate to labelled variables to the internal data array. Note that

- there may be up to 10000 variables and these are internally labeled V1, V2 $\cdots$ V10000 for transformation purposes. Values from the data file, ignoring any !SKIPed fields, are read into the leading variables,

- alpha (!A), integer (!I), pedigree (!P) and date (!DATE) fields are converted to real numbers (level codes) as they are read and before any transformations are applied,

- transformations may be applied to any variable (since every variable is numeric), but it may not be sensible to change factor level codes,
- transformations operate on a single variable (not a !G group of variables) unless it is explicitly stated otherwise,
- transformations are performed in order for each record in turn,
- variables that are created by transformation should be defined after (below) variables that are read from the data file unless it is the explicit intention to overwrite an input variable (see below),
- after completing the transformations for each record, the values in the record for variables associated with a label are held for analysis, (or the record (all values) is discarded; see !D transformation and Section 6.9),

Thus variables form three classes: those *read from the data file*, possibly modified, *and labelled* are available for subsequent use in analysis), those *created and labelled* are also available for subsequent use in the analysis and those *created or read but not labelled* (intermediate calculations) not required for subsequent analysis.

When listing variables in the field definitions, list those read from the data file first. After them, list (and define) the labelled variables that are to be created. The number of variables read can be explicitly set using the !READ qualifier described in Table 5.5. Otherwise, if the first transformation on a field overwrites its contents (for instance using !=), ASReml recognises that the field does not need to be read in (unless a subsequent field does need to be read). For example,

```
A
B
C !=A !-B
```

reads two fields (A and B), and constructs C as A-B. All three are available for analysis. However,

```
 A
B
C !=A !-B
D
E !=D !-B
```

reads four fields (A, B, C and D) because the fourth field is not obviously created and must therefore be read even though the third field (C) is overwritten. The fifth field is not read but just created E.

Variables that have an explicit label, may be referenced by their explicit label or their internal label. Therefore, to avoid confusion, do not use explicit labels of the form Vi, where i is a number, for variables to be referred to in a transformation. Vi always refers to field/variable i in a transformation statement.

Variables that are not initialized from the data file, are initialized to *missing value* for the first record, and otherwise, to the values from the preceding record (after transformation). Thus

```
A
B
LagA !=V4 !V4=A
```

reads two fields (`A` and `B`), and constructs `LagA` as the value of `A` from the previous record by extracting a value for `LagA` from working variable `V4` before loading `V4` with the current value of `A`.

5.5.1 Transformation syntax

Transformation qualifiers have one of seven forms, namely

`!`*operator*	to perform an operation on the current field, for example, `absY !ABS` to take absolute values,
`!`*operator value*	to perform an operation involving an argument on the current field, for example, `logY !=Y !^0` copies `Y` and then takes logs,
`!`*operator* `V`*field*	to perform an operation on the current field using the data in another field, for example, `!-V2` to subtract field 2 from the current field,
`!V`*target*	to reset the focus for subsequent transformations to field number *target*,
`!TARGET` *target*	to reset the focus for subsequent transformations to the previously named field *target*,
`!V`*target* `=` *value*	to set the target field to a particular value,
`!V`*target* `=` `V`*field*	to overwrite the data in a target field by the data values of another field; a special case is when *field* is `0` instructing ASReml to put the record number into the *target* field.

- *operator* is one of the symbols defined in Table 5.1,
- *value* is the argument, a real number, required by the transformation,
- `V` is the literal character and is followed by the number (*target* or *field*) of a data field; the data field is used or modified depending on the context,
- `V`*field* may be replaced by the label of the field if it already has a label,
- in the first three forms the operation is performed on the *current* field; this will be the field associated with the label unless the focus has been reset by specifying a new *target* in a preceding transformation,

- the last four forms change the focus of subsequent transformations to *target*,
- in the last two forms a value is assigned to the *target* field. For example, `... !V22=V11 ...` copies (existing) field 11 into field 22. Such a statement would typically be followed by more transformations. If there are fewer than 22 variables labelled then V22 is used in the transformation stage but not kept for analysis.
- only the `!DOM` and `!RESCALE` transformations automatically process a set of variables defined with the `!G` field definition. All other transformations always operate on only a single field. Use the `!DO ... !ENDDO` transformations to perform them on a set of variables.

Table 5.1: List of transformation qualifiers and their actions with examples

qualifier	argument	action	examples

`!=`	v	used to overwrite/create a variable with v. It usually implies the variable is not read (see examples on page 52)	`half !=0.5`
			`zero !=0.`

`!+`, `!-`, `!*`, `!/`	v	usual arithmetic meaning; note that, 0/0 gives 0 but $v/0$ gives a missing value where v is not 0.	`yield !/10`

`!^`	v	raises the data (which must be positive) to the power v.	`yield`
			`SQRyld !=yield`
			`!^0.5`

`!^`	0	takes natural logarithms of the data (which must be positive).	`yield`
			`LNyield !=yield`
			`!^0`

`!^`	-1	takes reciprocal of data (data must be positive).	`yield`
			`INVyield !=yield`
			`!^-1`

!>, !<, !<>, !==, !<=, !>=	v	logical operators forming 1 if true, 0 if false.	`yield`
			`high !=yield !>10`

`!ABS`		takes absolute values - no argument required.	`yield`
			`ABSyield !=yield`
			`!ABS`

`!ARCSIN`	v	forms an ArcSin transformation using the sample size specified in the argument, a number or another field. In the side example, for two existing fields `Germ` and `Total` containing counts, we form the ArcSin for their ratio (`ASG`) by copying the `Germ` field and applying the ArcSin transformation using the `Total` field as sample size.	`Germ Total`
			`ASG !=Germ !ARCSIN Total`

5.5 Transforming the data

Table 5.1: List of transformation qualifiers and their actions with examples

qualifier	argument	action	examples
!COS, !SIN	s	takes cosine and sine of the data variable with period s having default 2π; omit s if data is in radians, set s to 360 if data is in degrees.	Day CosDay !=Day !COS 365
!D, !D<>, !D<, !D<=, !D>, !D>=	v v v	!D[o] v discards records which have v or 'missing value' in the field, subject to the logical operator o.	yield !D<=0 yield !D<1 !D>100
!DV, !DV<>, !DV<, !DV<=, !DV>, !DV>=	v v v	!DV[o] v discards records, subject to the logical operator o, which have v in the field but keeps records with 'missing value' in the field; if !DV is used after !A or !I, v should refer to the encoded factor level rather than the value in the data file (see also Section 4.2). Use !DV * to discard just those records with a missing value in the field. !D v is equivalent to !DV * !DV v.	yield !DV<=0 yield !DV<1 !DV>100 InitialWt !DV *
!DO	$[n[i_t[i_v]]]$	causes ASReml to perform the following transformations n times (default is variables in current term), incrementing the target by i_t (default 1) and the argument (if present) by i_v (default 0). Loops may not be nested. A loop is terminated by !ENDDO, another !DO or a new field definition,	See below
!DOM	f	copies and converts additive marker covariables (-1, 0, 1) to dominance marker covariables (see below).	ChrAadd !G 10 !MM .. ChrAdom !DOM ChrAadd
!ENDDO		terminates a !DO transformation block	See below
!EXP		takes antilog base e - no argument required.	Rate !EXP

5.5 Transforming the data

Table 5.1: List of transformation qualifiers and their actions with examples

qualifier	argument	action	examples
!Jddm, !Jmmd !Jyyd		!Jddm converts a number representing a date in the form *ddmmccyy, ddmmyy* or *ddmm* into days. !Jmmd converts a date in the form *ccyymmdd, yymmdd* or *mmdd* into days. !Jyyd converts a date in the form *ccyyddd* or *yyddd* into days. These calculate the number of days since December 31 1900 and are valid for dates from January 1 1900 to December 31 2099; note that if *cc* is omitted it is taken as 19 if $yy > 32$ and 20 if $yy < 33$, the date must be entirely numeric: characters such as / may not be present (but see !DATE).	
!M, !M<>, !M< !M<= !M> !M>=	v v v	!Mv converts data values of v to missing; if !M is used after !A or !I, v should refer to the encoded factor level rather than the value in the data file (see also Section 4.2).	yield !M-9 yield !M<=0 !M>100
!MAX, !MIN, !MOD	v	the maximum, minimum and modulus of the field values and the value v.	yield !MAX 9
!MM	s	assigns Haldane map positions (s) to marker variables and imputes missing values to the markers (see below).	ChrAadd !G 10 !MM 1 ···
!NA	v	replaces any missing values in the variate with the value v. If v is another field, its value is copied.	Rate !NA 0 WT !=Wt2 !NA Wt1
!NORMAL	v	replaces the variate with normal random variables having variance v.	Ndat !=0 !Normal 4.5 is equivalent to Ndat !=Normal 4.5

!REPLACE	$o\ n$	replaces data values o with n in the current variable. I.e.
		IF(DataValue.EQ.o)
		DataValue=n
			Rate !REPLACE -9 0

!RESCALE	$o\ s$	rescales the column(s) in the current variable (!G group of variables) using $\boldsymbol{Y} = (\boldsymbol{Y} + o) * s$	Rate !RESCALE -10 0.1
!SEED	v	sets the seed for the random number generator.	··· !SEED 848586

Table 5.1: List of transformation qualifiers and their actions with examples

qualifier	argument	action	examples

`!SET`	*vlist*	for *vlist*, a list of n values, the data values $1 \ldots n$ are replaced by the corresponding element from *vlist*; data values that are < 1 or $> n$ are replaced by zero. *vlist* may run over several lines provided each incomplete line ends with a comma, i.e., a comma is used as a continuation symbol (see **Other examples** below).	`treat !L C A B`
			`CvR !=treat !SET 1 -1 -1`
			
			`group !=treat !SET 1,`
			`2 2 3 3 4`

`!SETN`	v n	`!SETN` v n replaces data values $1 : n$ with normal random variables having variance v. Data values outside the range $1 \cdots n$ are set to 0.	`Anorm !=A !SETN 2.5 10`
`!SETU`	v n	replaces data values $1 : n$ with uniform random variables having range $0 : v$. Data values outside the range $1 \cdots n$ are set to 0.	`Aeff !=A !SETU 5 10`
`!SUB`	*vlist*	replaces data values $= v_i$ with their index i where *vlist* is a vector of n values. Data values not found in *vlist* are set to 0. *vlist* may run over several lines if necessary provided each incomplete line ends with a comma. ASReml allows for a small rounding error when matching. It may not distinguish properly if values in *vlist* only differ in the sixth decimal place (see **Other examples** below).	`year 3 !SUB 66 67 68`
`!SEQ`		replaces the data values with a sequential number starting at 1 which increments whenever the data value changes between successive records; the current field is presumed to define a factor and the number of levels in the new factor is set to the number of levels identified in this sequential process (see **Other examples** below). Missing values remain missing.	`plot !=V3 !SEQ`

`!TARGET`	v	changes the focus of subsequent transformations to variable (field) v.	`sqrtA`
			`meanAB !+A !/2 ,`
			`!TARGET sqrtA !^0.5`

`!UNIFORM`	v	replaces the variate with uniform random variables having range $0 : v$.	`Udat !=0. !UNIFORM 4.5`

Table 5.1: List of transformation qualifiers and their actions with examples

qualifier	argument	action	examples
`!V`*target*=	*value*	assigns *value* to data field *target* overwriting previous contents; subsequent transformation qualifiers will operate on data field *target*.	··· `!V3=2.5`
	`V`*field*	assigns the contents of data field *field* to data field *target* overwriting previous contents; subsequent transformation qualifiers will operate on data field *target*. If *field* is 0 the number of the data record is inserted.	··· `!V10=V3` ··· `!V11=block` ··· `!V12=V0`

5.5.2 QTL marker transformations

`!MM` s associates marker positions in the vector s (based on the Haldane mapping function) with marker variables and replaces missing values in a vector of marker states with expected values calculated using distances to non-missing flanking markers. This transformation will normally be used on a `!G` n factor where the n variables are the marker states for n markers in a linkage group in map order and coded [-1,1] (backcross) or [-1,0,1] (F2 design). s (length n+1) should be the n marker positions relative to a left telomere position of zero, and an extra value being the length of the linkage group (the position of the right telomere). The length (right telomere) may be omitted in which case the last marker is taken as the end of the linkage group. The positions may be given in Morgans or centiMorgans (if the length is greater than 10, it will be divided by 100 to convert to Morgans).

The recombination rate between markers at s_L and s_R (L is left and R is right of some putative QTL at Q) is
$\theta_{LR} = (1 - e^{-2(s_R - s_L)})/2$.
Consequently, for 3 markers (L,Q,R), $\theta_{LR} = \theta_{LQ} + \theta_{QR} - 2\theta_{LQ}\theta_{QR}$.
The expected value of a missing marker at Q (between L and R) depends on the marker states at L and R: $E(q|1,1) = (1 - \theta_{LQ} - \theta_{QR})/(1 - \theta_{LR})$,
$E(q|1,-1) = (\theta_{QR} - \theta_{LQ})/\theta_{LR}$, $E(q|-1,1) = (\theta_{LQ} - \theta_{QR})/\theta_{LR}$
and $E(q|-1,-1) = (-1 + \theta_{LQ} + \theta_{QR})/(1 - \theta_{LR})$.
Let $\lambda_L = (E(q|1,1) + E(q|1,-1))/2 = \frac{\theta_{QR}(1-\theta_{QR})(1-2\theta_{LQ})}{\theta_{LR}(1-\theta_{LR})}$
and $\lambda_R = (E(q|-1,1) + E(q|-1,-1))/2 = \frac{\theta_{LQ}(1-\theta_{LQ})(1-2\theta_{QR})}{\theta_{LR}(1-\theta_{LR})}$
Then $E(q|x_L, x_R) = \lambda_L x_L + \lambda_R x_R$. Where there is no marker on one side, $E(q|x_R) = (1 - \theta_{QR})x_R + \theta_{QR}(-x_R) = x_R(1 - 2\theta_{QR})$. This qualifier facilitates the QTL method discussed in Gilmour (2007).

`!DOM` A is used to form dominance covariables from a set of additive marker covariables previously declared with the`!MM` marker map qualifier. It assumes the argument A is an existing group of marker variables relating to a linkage group defined using `!MM` which represents additive marker variation coded [-1, 0, 1] (representing marker states aa, aA and AA) respectively. It is a group transformation which takes the [-1,1] interval values, and calculates $(|X| - 0.5) * 2$ i.e. -1 and 1 become one, 0 becomes -1. The marker map is also copied and applied to this model term so it can be the argument in a `qtl()` term (page 100).

`!DO ... !ENDDO` provides a mechanism to repeat transformations on a set of variables. All tranformations except `!DOM and !RESCALE` operate once on a single field unless preceded by a `!DO` qualifier. The `!DO` qualifier has three arguments: $n[[i_t]i_v]$. n is the number of times the following transformations are to be performed. i_t (default 1) is the increment applied to the target field. i_v (default 0.0) is the increment applied to the transformation argument. The default for n is the number of variables in the current field definition. `!ENDDO` is formally equivalent to `!DO 1` and is implicit when another `!DO` appears or the next field definition begins. Note that when several transformations are repeated, the processing order is that each is performed n times before the next is processed (contrary to the implication of the syntax). However, the *target* is reset for each transformation so that the transformations apply to the same set of variables.

```
Y1 Y2 Y3 Y4 Y5           # Repeat 5 times, incrementing just
  Ymean !=0.  !DO 5 0 1 !+Y1 !ENDDO !/5 # the argument
```

is equivalent to

```
Y1 Y2 Y3 Y4 Y5
  Ymean !=0.  !+Y1 !+Y2 !+Y3 !+Y4 !+Y5 !/5
```

```
Y0 Y1 Y2 Y3 Y4 Y5 !TARGET Y1 !do 5 1 0 !-Y0 !ENDDO#Take Y0 from rest
Markers !G 12 !do !D * !ENDDO # Delete records with missing marker values
```

The default arguments (12, 1, 0.) are used. The initial target is the first marker.

5.5.3 Remarks concerning transformations

Note the following

- variables that are created should be listed after all variables that are read in unless the intention is to overwrite an input field.

- missing values are unaffected by arithmetic operations, that is, missing values in the current or target column remain missing after the transformation has been performed except in assignment
  - `!+3` will leave missing values (`NA`, `*` and `.`) as missing,
  - `!=3` will change missing values to `3`,

- multiple arithmetic operations cannot be expressed in a complex expression but must be given as separate operations that are performed in sequence as they appear, for example, `yield !-120 !*0.0333` would calculate `0.0333 * (yield - 120)`,

5.5 Transforming the data

- Most transformations only operate on a single field and will not therefore be performed on all variables in a `!G` factor set. The only transformations that apply to the whole set are `!DOM`, `!MM` and `!RESCALE`.

ASReml code	action
`yield !M0`	changes the zero entries in `yield` to missing values
`yield !^0`	takes natural logarithms of the `yield` data
`score !-5`	subtracts 5 from all values in `score`
`score !SET -0.5 1.5 2.5`	replaces data values of 1, 2 and 3 with -0.5, 1.5 and 2.5 respectively
`score !SUB -0.5 1.5 2.5`	replaces data values of -0.5, 1.5 and 2.5 with 1, 2 and 3 respectively; a data value of 1.51 would be replaced by 0 since it is not in the list or very close to a number in the list

`block 8`
`variety 20`
`yield`
`plot * !=variety !SEQ`
	in the case where
	– there are multiple units per plot,
	– contiguous plots have different treatments, and
	– the records are sorted units within plots within blocks,
	this code generates a `plot` factor assuming a new plot whenever the code in `V2` (`variety`) changes; whether this creates a variable or overwrites an input variable depends on whether any subsequent variables are input variables,

`Var 3`
`Nit 4`
`VxN 12 !=Var !-1 !*4 !+Nit`
	assuming `Var` is coded 1:3 and `Nit` is coded 1:4, this syntax could be used to create a new factor `VxN` with the 12 levels of the composite `Var` by `Nit` factor.

`YA !V98=YA !NA 0`
`YB !V99=YB !NA 0 !+V98 !D0`
	will discard records where **both** `YA` and `YB` have missing values (assuming neither have zero as valid data). The first line sets the focus to variable 98, copies `YA` into V98 and changes any *missing values* in `V98` to zero. The second line sets the focus to variable 99, copies `YB` into `V99` and changes any *missing values* in `V99` to zero. It then adds `V98` and discards the whole record if the result is zero, i.e. both `YA` and `YB` have missing values for that record. Variables `98` and `99` are not labelled and so are not retained for subsequent use in analysis.

5.5.4 Special note on covariates

Covariates are variates that appear as independent variables in the model. It is recommended that covariates be centred and scaled to have a mean near zero and a variance of about one to avoid failure to detect singularities. This can be achieved either

- externally to ASReml in data file preparation,

- using `!RESCALE` -*mean scale* where *mean* and *scale* are user supplied values, for example,
  `age !rescale -140 .142857 # in weeks`

5.6 Datafile line

The purpose of the datafile line is to

- nominate the data file,
- specify qualifiers to modify
  - the reading of the data,
  - the output produced,
  - the operation of ASReml.

```
NIN Alliance Trial 1989
 variety !A
⋮
 row 22
 column 11
nin89aug.asd !skip 1
yield ∼ mu variety
⋮
```

5.6.1 Data line syntax

The datafile line appears in the ASReml command file in the form

datafile [*qualifiers*]

- *datafile* is the path name of the file that contains the variates, factors, covariates, traits (response variates) and weight variables represented as data fields, see Chapter 4; enclose the path name in quotes if it contains embedded blanks,

- the *qualifiers* tell ASReml to modify either
  - the reading of the data and/or the output produced, see Table 5.2 below for a list of data file related qualifiers,

  - the operation of ASReml, see Tables 5.3 to 5.6 for a list of job control qualifiers

- the data file related qualifiers must appear on the data file line,

- the job control qualifiers may appear on the data file line or on following lines,

- the arguments to qualifiers are represented by the following symbols
  f — a filename,

  n — an integer number, typically a count, *p* — a vector of real numbers, typically in increasing order,

r — a real number,

s — a character string,

t — a model term label,

v — the number or label of a data variable,

vlist — a list of variable labels.

5.7 Data file qualifiers

Table 5.2 lists the qualifiers relating to data input. Use the Index to check for examples or further discussion of these qualifiers.

Table 5.2: Qualifiers relating to data input and output

qualifier	action
Frequently used data file qualifiers	
`!SKIP` *n*	causes the first *n* records of the (non-binary) data file to be ignored. Typically these lines contain column headings for the data fields.
Other data file qualifiers	
`!COLUMNFACTOR` *v* `!COLFAC` *v*	is used in combination with `!ROWFACTOR` [and `!SECTION`] to get ASReml to insert extra data records to complete the grid of plots defined by the *RowFactor* and the *ColumnFactor* for each *Section* so that a two-dimensional error structure can be defined (see `!SECTION` on page 73),
`!CSV`	used to make consecutive commas imply a missing value; this is auto-, matically set if the file name ends with `.csv` or `.CSV` (see Section 4.2) **Warning** This qualifier is ignored when reading binary data.
`!DATAFILE` *f*	specifies the datafile name replacing the one obtained from the datafile line. It is required when different `!PATHS` (see `!DOPATH` in Table 11.3) of a job must read different files. The `!SKIP` qualifier, if specified, will be applied when reading the file.
`!FILTER` *v* `[ !SELECT` *n*`] [ !EXCLUDE` *n*`]`	New R4 enables a subset of the data to be analysed; *v* is the number or name of a data field. When reading data, the value in field *v* is checked after any transformations are performed. If `!SELECT` and `!EXCLUDE` are omitted, records with zero in field *v* are omitted from the analysis. If `!SELECT` *n* is specified records with *n* in field *v* are retained and all other records are omitted. Conversely if `!EXCLUDE` *n* is specified, records with *n* in field *v* are ignored.

Table 5.2: Qualifiers relating to data input and output

qualifier	action

!FOLDER *s*	specifies an alternative folder for ASReml to find input files. This qualifier is usually placed on a separate line BEFORE the data filename line (and any pedigree/.giv .grm filename lines. For example,
	`!FOLDER ../Data`
	`data.asd !SKIP 1`
	is equivalent to
	`../Data/data.asd !SKIP 1`

!FORMAT *s*	supplies a Fortran like FORMAT statement for reading fixed format files. A simple example is !FORMAT(3I4,5F6.2) which reads 3 integer fields and 5 floating point fields from the first 42 characters of each data line. A format statement is enclosed in parentheses and may include 1 level of nested parentheses, for example, e.g. !FORMAT(4x,3(I4,f8.2)). Field descriptors are
	• rX to skip r character positions,
	• rAw to define r consecutive fields of w characters width,
	• rIw to define r consecutive fields of w characters width, and
	• rF$w.d$ to define r consecutive fields of w characters width; d indicates where to insert the decimal point if it is not explicitly present in the field,
	where r is an optional repeat count.
	In ASReml, the A and I field descriptors are treated identically and simply set the field width. Whether the field is interpreted alphabetically or as a number is controlled by the !A qualifier.
	Other legal components of a format statement are
	• the , character; required to separate fields - blanks are not permitted in the format.
	• the / character; indicates the next field is to be read from the next line. However a / on the end of a format to skip a line is not honoured.
	• BZ; the default action is to read blank fields as missing values. * and NA are also honoured as missing values. If you wish to read blank fields as zeros, include the string BZ.
	• the string BM; switches back to 'blank missing' mode.
	• the string Tc; moves the 'last character read' pointer to line position c so that the next field starts at position $c + 1$. For example T0 goes back to the beginning of the line.
	• the string D; invokes debug mode.
	A format showing these components is !FORMAT(D,3I4,8X,A6,3(2x,F5.2)/4x,BZ,20I1) and is suitable for reading 27 fields from 2 data records such as
	`111122223333xxxxxxxxALPHAFxx 4.12xx 5.32xx 6.32`
	`xxxx123 567 901 345 7890`

5.7 Data file qualifiers

Table 5.2: Qualifiers relating to data input and output

qualifier	action

`!MERGE` c f [`!SKIP` n] [`!MATCH` a b]	may be specified on a line following the datafile line. The purpose is to combine data fields from the (primary) data file with data fields from a secondary file (f). This `!MERGE` qualifier has been superseded by the much more powerful `MERGE` statement (see Chapter 12).
	The effect is to open the named file (skip n lines) and then insert the columns from the new file into field positions starting at position c. If `!MATCH` a b is specified, ASReml checks that the field a ($0 < a < c$) has the same value as field b. If not, it is assumed that the merged file has some missing records and missing values are inserted into the data record and the line from the MERGE file is kept for comparison with the next record.
	It is assumed that the lines in the MERGE file are in the same order as the corresponding lines occur in the primary data file, and that there are no extraneous lines in the MERGE file. A much more powerful merging facility is provided by the MERGE directive described in Chapter 12.
	For example, assuming the field definitions define 10 fields,
	` PRIMARY.DAT !skip 1`
	`!MERGE 6 SECOND.DAT !SKIP 1 !MATCH 1 6`
	would obtain the first five fields from `PRIMARY.DAT` and the next five from `SECOND.DAT`, checking that the first field in each file has the same value.
	Thus each input record is obtained by combining information from each file, before any transformations are performed.

`!READ` n	formally instructs ASReml to read n data fields from the data file. It is needed when there are extra columns in the data file that must be read but are only required for combination into earlier fields in transformations, or when ASReml attempts to read more fields than it needs to.
`!RECODE`	is required when reading a binary data file with pedigree identifiers that have not been recoded according to the pedigree file. It is not needed when the file was formed using the `!SAVE` option but will be needed if formed in some other way (see Section 4.2).
`!ROWFACTOR` v `!ROWFAC` v	is used in combination with `!COLUMNFACTOR` [and `!SECTION`] to get ASReml to insert extra data records to complete the grid of plots defined by the *RowFactor* and the *ColumnFactor* for each *Section* so that a two-dimensional error structure can be defined (see `!SECTION` on page 73),
`!RREC` [n]	causes ASReml to read n records or to read up to a data reading error if n is omitted, and then process the records it has. This allows data to be extracted from a file which contains trailing non-data records (for example extracting the predicted values from a `.pvs` file). The argument (n) specifies the number of data records to be read. If not supplied, ASReml reads until a data reading error occurs, and then processes the data it has. Without this qualifier, ASReml aborts the job when it encounters a data error. See `!RSKIP`.

Table 5.2: Qualifiers relating to data input and output

qualifier	action
`!RSKIP` n [s]	allows ASReml to skip lines at the heading of a file down to (and including) the nth instance of string s. For example, to read back the third set predicted values in a `.pvs` file, you would specify `!RREC !RSKIP 4 ' Ecode'` since the line containing the 4th instance of ' `Ecode`' immediately precedes the predicted values. The `!RREC` qualifier means that ASReml will read until the end of the predict table. The keyword `Ecode` which occurs once at the beginning and then immediately before each block of data in the `.pvs` file is used to count the sections.

5.7.1 Combining rows from separate files

ASReml can read data from multiple files provided the files have the same layout. The file specified as the 'primary data file' in the command file can contain lines of the form
`!INCLUDE` <filename> `!SKIP` n
where <filename> is the (path)name of the data subfile and `!SKIP` n is an optional qualifier indicating that the first n lines of the subfile are to be skipped. After reading each subfile, input reverts to the primary data file.

Typically, the primary data file will just contain `!INCLUDE` statements identifying the subfiles to include. For example, you may have data from a series of related experiments in separate data files for individual analysis. The primary data file for the subsequent combined analysis would then just contain a set of `!INCLUDE` statements to specify which experiments were being combined.

If the subfiles have CSV format, they should all have it and the `!CSV` file should be declared on the primary datafile line. This option is not available in combination with `!MERGE`.

5.8 Job control qualifiers

The following tables list the job control qualifiers. These change or control various aspects of the analysis. Job control qualifiers may be placed on the datafile line and following lines. They may also be defined using an environment variable called `ASREML_QUAL`. The environment variable is processed immediately after the datafile line is processed. All qualifier settings are reported in the `.asr` file. Use the Index to check for examples or further discussion of these qualifiers.

Important Many of these are only required in very special circumstances and new users should not attempt to understand all of them. You do need to understand that all general qualifiers are specified here. Many of these qualifiers are referenced in other chapters where their purpose will be more evident.

Table 5.3: List of commonly used job control qualifiers

qualifier	action

`!CONTINUE [`f`]`
`!MSV [`f`]`
`!TSV [`f`]`
	New R4 These qualifiers are used to restart/resume iterations from the point reached in a previous run. The qualifier `!CONTINUE [`f`]` can alternately be set from the command line using the option letter `C [`f`]` (see Section 11.3 on command line options). In each run ASReml writes the initial values of the variance parameters to a file with extension `.tsv` (template-start values) with information to identify individual variance parameters. After each iteration, ASReml writes the current values of the variance parameters to files with extension `.rsv` (re-start values) and `.msv`; the `.msv` version has information to clearly identify each variance parameter. If f is not set, then ASReml looks for a `.rsv` file with the same name used for the output files, ie. the `.as` name possibly appended by arguments. ASReml then scans this file for parameter values related to the current model, replacing the values obtained from the `.as` file before iteration resumes. If `!CONTINUE 2` or `!TSV` is used then the `.tsv` file is used instead of the `.rsv` file. Similarly, if `!CONTINUE 3` or `!MSV` are used then the `.msv` file is used instead of the `.rsv` file. If f=`filename`, with no extension, is used with `!CONTINUE`, `!TSV` or `!MSV`, ASReml will use the file `f.rsv` , `f.tsv` or `f.msv`. If f=`filename.xsv` with `x=r`, `t` or `m` is used with `!CONTINUE`, `!TSV` or `!MSV`, ASReml will use the file `f.xsv`. If the specified file is not present, ASReml reverts to reading the previous `.rsv` file. Some users may prefer, rather than specifying initial values in the model formulation, to generate a default `.tsv` file using `!MAXIT 0` and then edit the `.tsv` file with more appropriate values. If the model has changed and `!CONTINUE` is used, ASReml will pick up the values it recognises as being for the same terms from the `.rsv` file. Furthermore, `ASReml` will use estimates in the `.rsv` file for certain models to provide starting values for certain more general models, inserting reasonable defaults where necessary. The transitions recognised are listed and discussed in Section 7.9.2

Table 5.3: List of commonly used job control qualifiers

qualifier	action

`!CONTRAST` *s t p*	provides a convenient way to define contrasts among treatment levels. `!CONTRAST` lines occur as separate lines between the datafile line and the model line.
	s is the name of the model term being defined.
	t is the name of an existing factor.
	p is the list of contrast coefficients. For example
	`!CONTRAST LinN Nitrogen 3 1 -1 -3`
	defines `LinN` as a contrast based on the 4 (implied by the length of the list) levels of factor `Nitrogen`. Missing values in the factor become missing values in the contrast. Zero values in the factor (no level assigned) become zeros in the contrast. The user should check that the levels of the factor are in the order assumed by contrast (check the `.ass` or `.sln` or `.tab` files). It may also be used on the implicit factor `Trait` in a multivariate analysis provided it implicitly identifies the number of levels of `Trait`; the number of traits is implied by the length of the list. Thus, if the analysis involves 5 traits,
	`!CONTRAST Time Trait 1 3 5 10 20`

`!DDF` [*i*]	requests computation of the approximate denominator degrees of freedom according to Kenward and Roger (1997) for the testing of fixed effects terms in the dense part of the linear mixed model. There are three options for *i*: $i = -1$ suppresses computation, $i = 1$ and $i = 2$ compute the denominator d.f. using numerical and algebraic methods respectively. If *i* is omitted then $i = 2$ is assumed. If `!DDF` *i* is omitted, $i = -1$ is assumed except for small jobs ($<$ 10 parameters, $<$ 500 fixed effects, $<$ 10,000 equations and $<$ 100 Mbyte workspace) when $i = 2$. Calculation of the denominator degrees of freedom is computationally expensive. Numerical derivatives require an extra evaluation of the mixed model equations for every variance parameter. Algebraic derivatives require a large dense matrix, potentially of order number of equations plus number of records and is not available when `MAXIT` is 1 or for multivariate analysis.
`!FCON`	adds a 'conditional' Wald F statistic column to the Wald F Statistics table. It enables inference for fixed effects in the dense part of the linear mixed model to be conducted so as to respect both structural and intrinsic marginality (see Section 2.5). The detail of exactly which terms are conditioned on is reported in the `.aov` file. The marginality principle used in determining this conditional test is that a term cannot be adjusted for another term which encompasses it explicitly (e.g. term `A.C` cannot be adjusted for `A.B.C`) or implicitly (e.g. term `REGION` cannot be adjusted for `LOCATION` when locations are actually nested in regions although they are coded independently). `!FOWN` on page 78 provides a way of replacing the conditional Wald F statistic by specifying what terms are to be adjusted for, provided its degrees of freedom are unchanged from the incremental test.

5.8 Job control qualifiers

Table 5.3: List of commonly used job control qualifiers

qualifier	action
`!MAXIT` *n*	sets the maximum number of iterations; the default is 10 for traditional models, more for general models. ASReml iterates for n iterations unless convergence is achieved first. Convergence is presumed when the REML log-likelihood changes less than 0.002* *current iteration* number and the individual variance parameter estimates change less than 1%. If the job has not converged in n iterations, use the `!CONTINUE` qualifier to resume iterating from the current point. To abort the job at the end of the current iteration, create a file named `ABORTASR.NOW` in the directory in which the job is running. At the end of each iteration, ASReml checks for this file and if present, stops the job, producing the usual output but not producing predicted values since these are calculated in the last iteration. Creating `FINALASR.NOW` will stop ASReml after one more iteration (during which predictions will be formed). On case sensitive operating systems (eg. Unix), the filename (`ABORTASR.NOW` or `FINALASR.NOW`) must be upper case. Note that the `ABORTASR.NOW` file is deleted so nothing of importance should be in it. If you perform a system level abort (CTRL C or close the program window) output files other than the `.rsv` file will be incomplete. The `.rsv` file should still be functional for resuming iteration at the most recent parameter estimates (see `!CONTINUE`). Use `!MAXIT 1` where you want estimates of fixed effects and predictions of random effects for the particular set of variance parameters supplied as initial values. Otherwise the estimates and predictions will be for the updated variance parameters (see the `!BLUP` qualifier below). If `!MAXIT 1` is used and an Unstructured Variance model is fitted, ASReml will perform a Score test of the US matrix. Thus, assume the variance structure is modelled with reduced parameters, if that modelled structure is then processed as the initial values of a US structure, ASReml tests the adequacy of the reduced parameterization.
`!SUM`	causes ASReml to report a general description of the distribution of the data variables and factors and simple correlations among the variables for those records included in the analysis. This summary will ignore data records for which the variable being analysed is missing unless a multivariate analysis is requested or missing values are being estimated. The information is written to the `.ass` file.
`!X` *v* `!Y` *v* `!G` *v* `!JOIN`	is used to plot the (transformed) data. Use `!X` to specify the x variable, `!Y` to specify the y variable and `!G` to specify a grouping variable. `!JOIN` joins the points when the x value increases between consecutive records. The grouping variable may be omitted for a simple scatter plot. Omit `!Y` *y* produce a histogram of the x variable. For example, `!X age !Y height !G sex` Note that the graphs are only produced in the graphics versions of ASReml (Section 11.3).

Table 5.3: List of commonly used job control qualifiers

qualifier	action

	For multivariate repeated measures data, ASReml can plot the response profiles if the first response is nominated with the `!Y` qualifier and the following analysis is of the multivariate data. ASReml assumes the response variables are in contiguous fields and are equally spaced. For example
	`Response profiles`
	` Treatment !A`
	` Y1 Y2 Y3 Y4 Y5`
	`rat.asd !Y Y1 !G Treatment !JOIN`
	`Y1 Y2 Y3 Y4 Y5 ∼ Trait Treatment Trait.Treatment`

Table 5.4: List of occasionally used job control qualifiers

qualifier	action
`!ASMV` *n*	indicates a multivariate analysis is required although the data is presented in a univariate form. 'Multivariate Analysis' is used in the narrow sense where an unstructured error variance matrix is fitted across traits, records are independent, and observations may be missing for particular traits, see Chapter 8 for a complete discussion. The data is presumed arranged in lots of n records where n is the number of *traits*. It may be necessary to expand the data file to achieve this structure, inserting a missing value `NA` on the additional records. This option is sometimes relevant for some forms of repeated measures analysis. There will need to be a factor in the data to code for trait as the intrinsic `Trait` factor is undefined when the data is presented in a univariate manner.
`!ASUV`	indicates that a *univariate* analysis is required although the data is presented in a multivariate form. Specifically, it allows you to have an error variance other than $I \otimes \mathbf{\Sigma}$ where $\mathbf{\Sigma}$ is the unstructured (`US`, see Table 7.6) variance structure. If there are *missing values* in the data, include `!f mv` on the end of the linear model. The intrinsic factor `Trait` is defined and may be used in the model. See Chapter 8 for more information. This option is used for repeated measures analysis when the variance structure required is not the standard multivariate unstructured matrix.
`!DESIGN`	causes ASReml to write the design matrix, not including the response variable, to a `.des` file. It allows ASReml to create the design matrix required by the VCM process, see Section 7.8.2

5.8 Job control qualifiers

Table 5.4: List of occasionally used job control qualifiers

qualifier	action
!DISPLAY n	is used to select particular graphic displays. In spatial analysis of field trials, four graphic displays are possible (see Section 14.4). Coding these 1=variogram 2=histogram 4=row and column trends 8=perspective plot of residuals, set n to the sum of the codes for the desired graphics. The default is 9=1+8. These graphics are only displayed in versions of ASReml linked with Winteracter (that is, Linux, Mac and PC) versions. Line printer versions of these graphics are written to the .res file. See the G command line option (Section 11.3 on graphics) for how to save the graphs in a file for printing. Use !NODISPLAY to suppress graphic displays.
!EPS	sets hardcopy graphics file type to .eps.
!G v	is used to set a grouping variable for plotting, see !X.
!GKRIGE [p]	controls the expansion of !PVAL lists for fac(X,Y) model terms. For kriging prediction in 2 dimensions (X,Y), the user will typically want to predict at a grid of values, not necessarily just at data combinations. The values at which the prediction is required can be specified separately for X and Y using two !PVAL statements. Normally, predict points will be defined for all combinations of X and Y values. This qualifier is required (with optional argument 1) to specify the lists are to be taken in parallel. The lists must be the same length if to be taken in parallel. Be aware that adding two dimensional prediction points is likely to substantially slow iterations because the variance structure is dense and becomes larger. For this reason, ASReml will ignore the extra PVAL points unless either !FINAL or !GKRIGE are set, to save processing time.
!GROUPFACTOR t v p	The !GROUPFACTOR qualifier, like !SUBSET, must appear on a line by itself after the data line and before the model line. Its purpose is to define a factor t by merging levels of an existing factor v. The syntax is !GROUPFACTOR <*Group_factor*> <*Exist_factor*> <*new codes*> for example !GROUPFACTOR Year YearLoc 1 1 1 2 2 3 3 3 4 4 forms a new factor Year with 4 levels from the existing factor YearLoc with 10 levels. Alternatively, Year could be formed by data transformation: Year * !=YearLoc !set 1 1 1 2 2 3 3 3 4 4 !L 2001 2002 2003 2004
!IDLIMIT v	is used when ASReml expands a residual statement like residual sat(Site).ar1(row).ar1(col) and the dimension of row or col is small. The ar1() structure is changed to id(). When the number of rows/columns is less than or equal to v, the structure is set to ID instead of AR1. v has a default value of 4 and cannot be reset to less than 3. If the qualifier is not specified the value of v is 1.
!JOIN	is used to join lines in plots, see !X.

5.8 Job control qualifiers

Table 5.4: List of occasionally used job control qualifiers

qualifier	action

!MBF mbf(v,n) f
[!FACTOR]
[!FIELD s]
[!KEY k]
[!NOKEY]
[!RENAME t]
[!RFIELD r]
[!SKIP k]
[!SPARSE]
	specified on a separate line after the datafile line predefines the model term mbf(v,n) as a set of n covariates indexed by the data values in variable v. MBF stands for My Basis Function and uses the same mechanism as the leg(), pol() and spl() model functions but with covariates supplied by the user. It is used for reading in specialized design matrices indexed by a factor in the data including genetic marker covariables. By default, the file f should contain 1+n fields where the first field, the *key* field, contains the values which are in the data variable or at which prediction is required, and the remaining n fields define the corresponding covariate values. If n is omitted, all fields after the *key* field, are taken unless !FACTOR is specified for which n is 1 and the covariate values are treated as coding for a multilevel factor. Set n to 1 to read just one field form the data file. Also note that the file may be a binary file (e.g. formed in a previous run using !SAVE). !RENAME t changes the name of the the term from mbf(...) to the new name t. This is necessary when several mbf(...) terms are being defined which would otherwise have the same name/label. For example
	`!MBF mbf(entry) mlib/m35.csv !RENAME Marker35`
	If the *key* values are the ordered sequence $1 : N$, the *key* field may be omitted if !NOKEY is specified. If the *key* is not in the first field, its location can be specified with !KEY k. If extracting a single covariate from a large set of covariates in the file, the specific field to extract can be given by !FIELD s in absolute terms, or relative to the *key* field by !RFIELD r. For example
	`!MBF mbf(variety,1) markers.csv !key 1 !RFIELD 35 !RENAME Marker35`
	!SKIP k requests the first k lines of the file be ignored.
	!SPARSE can be used when the covariates are predominately zero. Each *key* value is followed by as many *column,value* pairs as required to specifiy the non zero elements of the design for that value of *key*. The pairs should be arranged in increasing order of *column* within rows. The rows may be continued on subsequent lines of the file provided incomplete lines end with a COMMA.
	This file may now be a binary format file, with file extension .bin indicating 32bit real binary numbers and .dbl indicating 64bit real binary values. Files with these formats can be easily created in a preliminary run using the !SAVE qualifier. The advantage of using a binary file is that reading the file is much quicker. This is important if the file has many fields and is being accessed repeatedly, for example
	`!CYCLE 1:1000`
	`!MBF mbf(Geno) markers.dbl !key 1 !RFIELD $I !rename M$I`
	`... !r M$I`

Table 5.4: List of occasionally used job control qualifiers

qualifier	action
	Restrictions: The *key* field MUST be numeric. In particular, if the data field it relates to is either an `!A` or `!I` encoded factor, the original (uncoded) level labels may not specified in the MBF file. Rather the coded levels must be specified. The MBF file is processed before the data file is read in and so the mapping to coded levels has not been defined in ASReml when the MBF file is processed, although the user can/must anticipate what it will be. Comment: If this MBF process is to be used repeatedly, for example to process a large set of marker variables in conjunction with `!CYCLE`, processing will be much faster if the markers variables are in separate files. ASReml will read 10 files containing a single field much faster than reading a single file containing 400 fields, ten times to extract 10 different markers. Also note that the file may be a binary file and will be read much quicker than a formatted file. A binary file may be formed in a previous run using `!SAVE`.
`!MVINCLUDE`	When missing values occur in the design ASReml will report this fact and abort the job unless `!MVINCLUDE` is specified (see Section 6.9); then missing values are treated as zeros. Use the `!DV` transformation to drop the records with the missing values.
`!MVREMOVE`	instructs ASReml to discard records which have missing values in the design matrix (see Section 6.9).
`!NODISPLAY`	suppresses the graphic display of the variogram and residuals which is otherwise produced for spatial analyses in the PC version. This option is usually set on the command line using the option letter `N` (see Section 11.3 on graphics). The text version of the graphics is still written to the `.res` file.
`!PVAL` *v p*	is a mechanism for specifying the particular points to be predicted for covariates modelled using `fac(`*v*`)`, `leg(`*v*`,`*k*`)`, `spl(`*v*`,`*k*`)` and `pol(`*v*`,`*k*`)`. The points are specified here so that they can be included in the appropriate design matrices. *v* is the name of a data field. *p* is the list of values at which prediction is required. See `!GKRIGE` for special conditions pertaining to `fac(`*x*`,`*y*`)` prediction.
`!PVAL` *f vlist*	is used to read *predict_points* for several variables from a file *f*. *vlist* is the names of the variables having values defined. If the file contains unwanted fields, put the pseudo variate label `skip` in the appropriate position in *vlist* to ignore them. The file should only have numeric values. *predict_points* cannot be specified for design factors.

Table 5.4: List of occasionally used job control qualifiers

qualifier — action

`!SECTION` *v*

specifies the variable in the data that defines the data sections. This qualifier enables ASReml to check that sections have been correctly dimensioned. Further, when the model term `mv` is included in the model and `!ROWFACTOR` and `!COLUMNFACTOR` are defined, ASReml will check that the observations in each section form a complete grid; if not the grid will be completed by adding the appropriate extra data records. If only one grid is required from all the data then the `!SECTION` variable does not need specifying. The following is a basic example assuming 5 sites (sections).

```
Basic multi-envt trial analysis filling out row and column
grid
 site   5   # sites coded 1 ...  5
 column *   # columns coded 1 ...
 row    *   # rows coded 1 ...
 variety !A # variety names
 yield
met.dat !SECTION site !ROWFACTOR row !COLUMNFACTOR col
yield ~ site !r variety site.variety !f mv
residual sat(site).ar1(row).ar1(column)
```

`!SPLINE spl(`*v*`,`*n*`)` *p*

defines a spline model term with an explicit set of knot points. The basic form of the spline model term, `spl(`*v*`)`, is defined in Table 6.1 where *v* is the underlying variate. The basic form uses the unique data values as the knot points. The extended form is `spl(`*v*`,`*n*`)` which uses *n* knot points. Use this `!SPLINE` qualifier to supply an explicit set of *n* knot points (*p*) for the model term *t*. Using the extended form without using this qualifier results in *n* equally spaced knot points being used. The `!SPLINE` qualifier may only be used on a line by itself after the datafile line and before the model line.

When knot points are explicitly supplied they should be in increasing order and adequately cover the range of the data or ASReml will modify them before they are applied. If you choose to spread them over several lines use a comma at the end of incomplete lines so that ASReml will to continue reading values from the next line of input. If the explicit points do not adequately cover the range, a message is printed and the values are rescaled unless `!NOCHECK` is also specified. Inadequate coverage is when the explicit range does not cover the midpoint of the actual range. See `!KNOTS`, `!PVAL` and `!SCALE`.

`!STEP` *r*

reduces the update step sizes of the variance parameters. The default value is the reciprocal of the square root of `!MAXIT`. It may be set between 0.01 and 1.0. The step size is increased towards 1 each iteration. Starting at 0.1, the sequence would be 0.1, 0.32, 0.56, 1. This option is useful when you do not have good starting values, especially in multivariate analyses.

`!SUBGROUP` *t v p*

forms a new group factor (*t*) derived from an existing group factor (*v*) by selecting a subset (*p*) of its variables. A subgroup factor may not be used in a `PREDICT` or `TABULATE` directive.

Table 5.4: List of occasionally used job control qualifiers

qualifier	action

`!SUBSET` *t v p*	forms a new factor (t) derived from an existing factor (v) by selecting a subset (p) of its levels. Missing values are transmitted as missing and records whose level is zero are transmitted as zero. The qualifier occupies its own line after the datafile line but before the linear model. e.g.
	`!SUBSET EnvC Env 3 5 8 9 :15 21 33`
	defines a reduced form of the factor Env just selecting the environments listed. It might then be used in the model in an interaction. A subset factor can be used in a `TABULATE` directive but not in a `PREDICT` directive.
	The intention is to simplify the model specification in MET (Multi Environment Trials) analyses where say Column effects are to be fitted to a subset of environments. It may also be used on the intrinsic factor `Trait` in a multivariate analysis provided it correctly identifies the number of levels of `Trait` either by including the last trait number, or appending sufficient zeros. Thus, if the analysis involves 5 traits,
	`!SUBSET Trewe Trait 1 3 4 0 0`

`!WMF`	sets hardcopy graphics file type to `.wmf`.

Table 5.5: List of rarely used job control qualifiers

qualifier	action
`!AILOADINGS` *i*	controls modification to AI updates of loadings in extended Factor Analytic models. After ASReml calculates updates for variance parameters, it checks whether the updates are reasonable and sometimes reduces them over and above any `!STEPSIZE` 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 back. Unless the user gives constraints, ASReml sets them and rotates the loadings each iteration. When `!AILOADINGS` i is specified, 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. If `!AILOADINGS` is not specified and `!CONTINUE` is used and initializes the `XFA` model from a lower order, the i parameter is set internally.
`!AISINGULARITIES`	can be specified to force a job to continue even though a singularity was detected in the Average Information (AI) matrix. The AI matrix is used to give updates to the variance parameter estimates. In release 1, if singularities were present in the AI matrix, a generalized inverse was used which effectively conditioned on whichever parameters were identified as singular. ASReml now aborts processing if such singularities appear unless the `!AISINGULARITIES` qualifier is set. Which particular parameter is singular is reported in the variance component table printed in the `.asr` file.

Table 5.5: List of rarely used job control qualifiers

qualifier	action
	The most common reason for singularities is that the user has overspecified the model and is likely to misinterpret the results if not fully aware of the situation. Overspecification will occur in a direct product of two unconstrained variance matrices (see Section 7.4), when a random term is confounded with a fixed term and when there is no information in the data on a particular component. Another common cause is when fitting an animal model and there is excessive sire/dam variance (so that heritability from a sire model would exceed 1) so that the residual variance under the animal model has approached zero. In this case the data contradicts the assumptions of the animal model. The best solution is to reform the variance model so that the ambiguity is removed, or to fix one of the parameters in the variance model so that the model can be fitted. Only rarely will it be reasonable to specify the `!AISINGULARITIES` qualifier.
`!BMP`	sets hardcopy graphics file type to `.bmp`.
`!BRIEF` [n]	suppresses some of the information written to the `.asr` file. The data summary and regression coefficient estimates are suppressed. This qualifier should not be used for initial runs of a job until the user has confirmed from the data summary that the data is correctly interpreted by ASReml. Use `!BRIEF` 2 to cause the predicted values to be written to the `.asr` file instead of the `.pvs` file. Use `!BRIEF` -1 to get BLUE (fixed effect) estimates reported in `.asr` file. The `!BRIEF` qualifier may be set with the `B` command line option.
`!BLUP` n	is used to calculate the effects reported in the `.sln` file without calculating any derived quantities such as predicted values or updated variance parameters. For argument values 1:3, ASReml solves for the effects directly while for values 4:19 it solves the mixed model equations by iteration, allowing larger models to be fitted. With direct solution, the estimation REML iteration routine is aborted after $n = 1$: forming the estimates of the vector of fixed and random effects by matrix inversion, $n = 2$: forming the estimates of the vector of fixed and random effects, REML log-likelihood and residuals (this is the default), $n = 3$: forming the estimates of the vector of fixed and random effects, REML log-likelihood, residuals and inverse coefficient matrix. For arguments 4, 10:19, ASReml forms the mixed model equations and solves them iteratively to obtain solutions for the fixed and random effects. The options are: $n = 4$: forming the estimates of the vector of fixed and random effects using the Preconditioned Conjugate Gradient (PCG) Method (Mrode, 2005),

Table 5.5: List of rarely used job control qualifiers

qualifier	action
	n = 10:19 forming the estimates of the vector of fixed and random effects by Gauss-Seidel iteration of the mixed model equations, with relaxation factor $n/10$, The default maximum number of iterations is 12000. This can be reset by supplying a value greater than 100 with the `!MAXIT` qualifier in conjunction with the `!BLUP` qualifier. Iteration stops when the average squared update divided by the average squared effect is less than $1e^{-10}$. Gauss-Seidel iteration is generally much slower than the PCG method. ASReml prints its standard reports as if it had completed the iteration normally, but since it has not completed it, some of the information printed will be incorrect. In particular, variance information on the variance parameters will always be unavailable. Standard errors on the estimates will be wrong unless n=3. Residuals are not available if n=1. Use of n=3 or n=2 will halve the processing time when compared to the alternative of using `!MAXIT 1` rather than a !tt !BLUP n qualifier. However, `!MAXIT 1` does result in complete and correct output.
`!DENSE` n	sets the number of equations solved densely up to a maximum of 5000. By default, sparse matrix methods are applied to the random effects and any fixed effects listed after random factors or whose equation numbers exceed 800. Use `!DENSE` n to apply sparse methods to effects listed before the `!r` (reducing the size of the DENSE block) or if you have large fixed model terms and want Wald F statistics calculated for them. Individual model terms will not be split so that only part is in the dense section. n should be kept small (<100) for faster processing.
`!DF` n	alters the error degrees of freedom from ν to $\nu+n$. This qualifier might be used when analysing pre-adjusted data to reduce the degrees of freedom (n negative) or when weights are used in lieu of actual data records to supply error information (n positive). The degrees of freedom is only used in the calculation of the residual variance in a univariate single site analysis. The option will have no effect in analyses with multiple error variances (for sites or traits) other than in the reported degrees of freedom. Use `!ADJUST` r rather than `!DF` n if r is not a whole number. Use with `!YSS` r to supply variance when data fully fitted.

Table 5.5: List of rarely used job control qualifiers

qualifier	action
!EMFLAG *n* !PXEM *n*	requests ASReml use Expectation-Maximization (EM) rather than Average Information (AI) updates when the AI updates would make a US structure non-positive definite. This only applies to US structures and is still under development. When !GP is associated with a US structure, ASReml checks whether the updated matrix is positive definite (PD). If not, it replaces the AI update with an EM update. If the non PD characteristic is transitory, then the EM update is only used as necessary. If the converged solution would be non PD, there will be a EM update each iteration even though !EM is omitted. EM is notoriously slow at finding the solution and ASReml includes several modified schemes, discussed by Cullis *et al.* (2004), particularly relevant when the AI update is consistently outside the parameter space. These include optionally performing extra local EM or PXEM (Parameter Expanded EM) iterates. These can dramatically reduce the number of iterates required to find a solution near the boundary of the parameter space but do not always work well when there are several matrices on the boundary. The options are !EMFLAG [1] Standard EM plus 10 local EM steps !EMFLAG 2 Standard EM plus 10 local PXEM steps !PXEM [2] Standard EM plus 10 local PXEM steps !EMFLAG 3 Standard EM plus 10 local EM steps !EMFLAG 4 Standard EM plus 10 local EM steps !EMFLAG 5 Standard EM only !EMFLAG 6 Single local PXEM !EMFLAG 7 Standard EM plus 1 local EM step !EMFLAG 8 Standard EM plus 10 local EM steps Options 3 and 4 cause all US structures to be updated by (PX)EM if any particular one requires EM updates. The test of whether the AI updated matrix is positive definitite is based on absorbing the matrix to check all pivots are positive. Repeated EM updates may bring the matrix closer to being singular. This is assessed by dividing the pivot of the first element with the first diagonal element of the matrix. If it is less than 10^{-7} (this value is consistent with the multiple partial correlation of the first variable with the rest being greater than 0.9999999, ASReml fixes the matrix at that point and estimates any other parameters conditional on these values. To preceed with further iterations without fixing the matrix values would ultimately make the matrix such that it would be judged singular resulting the analysis being aborted.

Table 5.5: List of rarely used job control qualifiers

qualifier	action
`!EQORDER` *o*	modifies the algorithm used for choosing the order for solving the mixed model equations. A new algorithm devised for release 2 is now the default and is formally selected by `!EQORDER 3`. The algorithm used for release 1 is essentially that selected by `!EQORDER 1`. The new order is generally superior. `!EQORDER -1` instructs ASReml to process the equations in the order they are specified in the model. Generally this will make a job much slower, if it can run at all. It is useful if the model has a suitable order as in the IBD model `Y ~ mu !r !{ giv(id) id !}` `giv(id)` invokes a dense inverse of an IBD matrix and `id` has a sparse structured inverse of an additive relationship matrix. While `!EQORDER 3` generates a more sparse solution, `!EQORDER -1` runs faster.
`!EXTRA` *n*	forces another `mod(`*n*`,10)` rounds of iteration after apparent convergence. The default for *n* is 1. This qualifier has lower priority than `!MAXIT` and `ABORTASR.NOW` (see `!MAXIT` for details). Convergence is judged by changes in the REML log-likelihood value and variance parameters. However, sometimes the variance parameter convergence criteria has not been satisfied.

`!FOWN`	allows the user to specify the test reported in the `F-con` column of the Wald F Statistics table. It has the form
	`!FOWN` *terms to test* ; *background terms*
	placed on a separate line immediately after the model line. Multiple `!FOWN` statements should appear together. It generates a Wald F statistic for each model term in *terms to test* which tests its contribution after all other terms in *terms to test* and *background terms*, conditional on all terms that appear in the SPARSE equations. It should only specify terms which will appear in the table of Wald F statistics.
	
	For example,
	`!FOWN A B C ; mu`
	`!FOWN A.B B.C A.C ; mu A B C`
	`!FOWN A.B.C ; mu A B C A.B B.C A.C`
	would request the Wald F statistics based on (see page 19)
	R(`A` | `mu B C` *sparse*),
	R(`B` | `mu A C` *sparse*),
	R(`C` | `mu A B` *sparse*),
	R(`A.B` | `mu A B C B.C A.C` *sparse*),
	R(`B.C` | `mu A B C A.B A.C` *sparse*),
	R(`A.C` | `mu A B C A.B B.C` *sparse*) and
	R(`A.B.C` | `mu A B C A.B A.C B.C` *sparse*).

Table 5.5: List of rarely used job control qualifiers

qualifier	action
	Warnings: • For computational convenience, ASReml calculates `!FOWN` tests using a full rank parameterization of the fitted model with rank (numerator degrees of freedom, NumDF) of terms generated by the incremental Wald F tests. • Unfortunately, if some terms in the implicit model defined by the requested `!FOWN` test would have more or less NumDF than are present in the full rank parameterization because aliased effects are reordered, it can not be calculated correctly from the full rank parameterization. In this case ASReml reverts to the 'conditional' test but identifies the terms that need to be reordered in the fitted model to obtain the `!FOWN` test(s) specified. It is necessary to rerun ASReml after reordering these terms to obtain the `!FOWN` test(s) specified. Several reruns may be needed to perform all `!FOWN` tests specified. • Any model terms in the `!FOWN` lists which do not appear in the actual model, are ignored without flagging an error. • Any model terms which are omitted from `!FOWN` statements are tested with the usual conditional test. • If any model terms are listed twice, only the first test is performed. F-con tests specified in `!FOWN` statements are given model codes `O`, `P`, ... The `!FOWN` statements are parsed by the routine that parses the model line and so accepts the same model syntax options. Care should be taken to ensure term names are spelt exactly as they appear in the model.
`!GDENSE`	is used to have the first random term included in the *dense* equations if it is a `GRM/GIV` variance structure. This will result in faster processing when the GRM (inverse) matrix is not sparse.
`!GLMM` [n]	sets the number of inner iterations performed when a iteratively weighted least squares analysis is performed. Inner iterations are iterations to estimate the effects in the linear model for the current set of variance parameters. Outer iterations are the AI updates to the variance parameters. The default is to perform 4 inner iterations in the first round and 2 in subsequent rounds of the outer iteration. Set n to 2 or more to increase the number of inner iterations.
`!HPGL [2]`	sets hardcopy graphics file type to HP GL. An argument of 2 sets the hardcopy graphics file type to HP GL 2
`!HOLD` [*list*]	allows the user to temporarily fix the parameters listed. Parameter numbers have been added to the reporting of input values to facilitate use of this and other parameter number dependent qualifiers. The list should be in increasing order using colon to indicate a sequence, step size is 1. For example `!HOLD 1:20 30:40` .

Table 5.5: List of rarely used job control qualifiers

qualifier	action

`!LAST` <factor$_1$ > <lev$_1$ > [<fac$_2$ > <lev$_2$ > <fac$_3$ > <lev$_3$ >]	limits the order in which equations are solved in ASReml by forcing equations in the sparse partition involving the first <lev$_i$ > equations of <factor$_i$ > to be solved after all other equations in the sparse partition. 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. The test example had
	`!r Anim Litter !f HYS`
	where genetic groups were included in the definition of `Anim`.
	
	Consequently, there were 5 singularities in `Anim`. The default reordering allows those singularities to appear anywhere in the `Anim` and `HYS` terms. Since 29 genetic groups were defined in `Anim`, `!LAST Anim 29` forces the genetic group equations to be absorbed last (and therefore incorporate any singularities). In the more general model fitting
	`!r Tr.Anim Tr.Lit !f Tr.HYS`
	without `!LAST`, the location of singularities will almost surely change if the G structures for `Tr.Anim` or `Tr.Lit` are changed, invalidating Likelihood Ratio tests between the models.

`!OUTLIER`	performs the outlier check described on page 17. This can have a large time penalty in large models.
`!OWN` *f*	supplies the name of a program supplied by the user in association with the `OWN` variance model (page 127).
`!PRINT` *n*	causes ASReml to print the transformed data file to *basename*`.asp`. If $n < 0$, data fields `1...mod(`*n*`)` are written to the file, $n = 0$, nothing is written, $n = 1$, all data fields are written to the file if it does not exist, $n = 2$, all data fields are written to the file overwriting any previous contents, $n > 2$, data fields $n \ldots t$ are written to the file where t is the last defined column.
`!PNG`	sets hardcopy graphics file type to `.png`.
`!PS`	sets hardcopy graphics file type to `.ps`.

`!PVSFORM` n	modifies the format of the tables in the `.pvs` file and changes the file extension of the file to reflect the format.
	`!PVSFORM` 1 is TAB separated: `.pvs` → `_pvs.txt`
	`!PVSFORM` 2 is COMMA separated: `.pvs` → `_pvs.csv`
	`!PVSFORM` 3 is Ampersand separated: `.pvs` → `_pvs.tex`
	See `!TXTFORM` for more detail.

`!RESIDUALS [2]`	instructs ASReml to write the transformed data and the residuals to a binary file. The residual is the last field. The file *basename*`.srs` is written in single precision unless the argument is 2 in which case *basename*`.drs` is written in double precision. Factor names are held in a `.vll` file: see `!SAVE` below.

Table 5.5: List of rarely used job control qualifiers

qualifier	action
	The file will not be written from a spatial analysis (two-dimensional error) when the data records have been sorted into field order because the residuals are not in the same order that the data is stored. The residual from a spatial analysis will have the `units` part added to it when `units` is also fitted. The `.drs` file could be renamed (with extension `.dbl`) and used for input in a subsequent run.
`!SAVE` *n*	instructs ASReml to write the data to a binary file. The file `asrdata.bin` is written in single precision if the argument *n* is `1` or `3`; `asrdata.dbl` is written in double precision if the argument *n* is `2` or `4`; the data values are written before transformation if the argument is `1` or `2` and after transformation if the argument is `3` or `4`. The default is single precision after transformation (see Section 4.2). When either `!SAVE` or `!RESIDUALS` is specified, ASReml saves the factor level labels to a *basename*`.vll` and attempts to read them back when data input is from a binary file. Note that if the job `basename` changes between runs, the `.vll` file will need to be copied to the new *basename*. If the `.vll` file does not match the factor structure (i.e. the same factors in the same order), reading the `.vll` file is aborted.
`!SCREEN` [*n*] [`!SMX` *m*]	performs a 'Regression Screen', a form of all subsets regression. For *d* model terms in the DENSE equations, there are 2^d-1 possible submodels. Since for $d > 8$, $2^d - 1$ is large, the submodels explored are reduced by the parameters *n* and *m* so that only models with at least *n* (default 1) terms but no more than *m* (default 6) terms are considered. The output (see page 222) is a report to the `.asr` file with a line for every submodel showing the sums of squares, degrees of freedom and terms in the model. There is a limit of $d = 20$ model terms in the screen. ASReml will not allow interactions to be included in the screened terms. For example, to identify which three of my set of 12 covariates best explain my dependent variable given the other terms in the model, specify `!SCREEN 3 !SMX 3`. The number of models evaluated quickly increases with *d* but ASReml has an arbitrary limit of 900 submodels evaluated. Use the `!DENSE` qualifier to control which terms are screened. The screen is conditional on all other terms (those in the SPARSE equations) being present.

`!SLNFORM` [*n*]	modifies the format of the `.sln` file.
	`!SLNFORM -1` prevents the `.sln` file from being written.
	`!SLNFORM 1` is TAB separated: `.sln` becomes `_sln.txt`
	`!SLNFORM 2` is COMMA separated: `.sln` becomes `_sln.csv`
	`!SLNFORM 3` is Ampersand separated: `.sln` becomes `_sln.tex`
	Note that, extra signifcant digits are reported when `!SLNFORM` is set, and expanded labelling of the levels in interactions is used because field width is no longer restricted. See `!TXTFORM` for more detail.

`!SPATIAL`	increases the amount of information reported on the residuals obtained from the analysis of a two dimensional regular grid field trial. The information is written to the `.res` file.

Table 5.5: List of rarely used job control qualifiers

qualifier	action
!TABFORM [n]	controls form of the .tab file !TABFORM 1 is TAB separated: .tab becomes _tab.txt !TABFORM 2 is COMMA separated: .tab becomes _tab.csv !TABFORM 3 is Ampersand separated: .tab becomes _tab.tex See !TXTFORM for more detail.
!TXTFORM [n]	sets the default argument for !PVSFORM, !SLNFORM, !TABFORM and !YHTFORM if these are not explicitly set. !TXTFORM (or !TXTFORM 1) replaces multiple spaces with TAB and changes the file extension to, say, _sln.txt. This makes it easier to load the solutions into Excel. !TXTFORM 2 replaces multiple spaces with COMMA and changes the file extension to, say, _sln.csv. However, since factor labels sometimes contain COMMAS, this form is not so convenient. !TXTFORM 3 replaces multiple spaces with Ampersand, appends a double backslash to each line and changes the file extension to say _sln.tex (Latex style). Additional significant digits are reported with these formats. Omitting the qualifier means the standard fixed field format is used. For .yht and .sln files, setting n to -1 means the file is not formed.
!TWOWAY	modifies the appearance of the variogram calculated from the residuals obtained when the sampling coordinates of the spatial process are defined on a lattice. The default form is based on absolute 'distance' in each direction. This form distinguishes same sign and different sign distances and plots the variances separately as two layers in the same figure.
!VCC n	specifies that n constraints are to be applied to the variance parameters. The constraint lines occur after the G structures are defined. The constraints are described in Section 7.8.2. The variance header line (structural specification) or residual line (Section 6.2) must be present, even if only 0 0 0 or residual units indicating there are no explicit R or G structures (see Section 7.8.2).
!VGSECTORS [s]	requests that the variogram formed with radial coordinates (see page 18) be based on s (4, 6 or 8) sectors of size $180/s$ degrees. The default is 4 sectors if !VGSECTORS is omitted and 6 sectors if it is specified without an argument. The first sector is centred on the X direction. Figure 5.1 is the variogram using radial coordinates obtained using predictors of random effects fitted as fac(xsca,ysca). It shows low semivariance in xsca direction, high semivariance in the ysca direction with intermediate values in the 45 and 135 degrees directions.
!YHTFORM [f]	controls the form of the .yht file !YHTFORM -1 suppresses formation of the .yht file !YHTFORM 1 is TAB separated: .yht becomes _yht.txt !YHTFORM 2 is COMMA separated: .yht becomes _yht.csv !YHTFORM 3 is Ampersand separated: .yht becomes _yht.tex
!YSS [r]	adds r to the total Sum of Squares. This might be used with !DF to add some variance to the analysis when analysing summarised data.

Table 5.5: List of rarely used job control qualifiers

qualifier	action

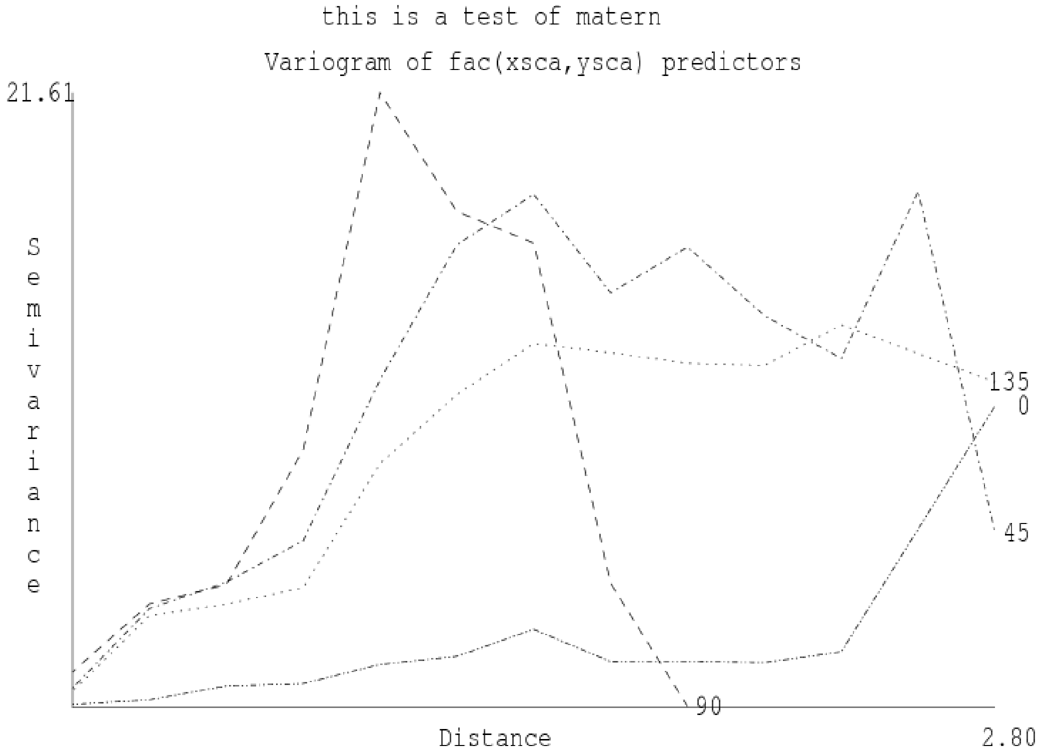

Figure 5.1: Variogram in 4 sectors for Cashmore data

Table 5.6: List of very rarely used job control qualifiers

qualifier	action
`!CINV` *n*	prints the portion of the inverse of the coefficient matrix pertaining to the n^{th} term in the linear model. Because the model has not been defined when ASReml reads this line, it is up to the user to count the terms in the model to identify the portion of the inverse of the coefficient matrix to be printed. The option is ignored if the portion is not wholly in the `SPARSE` stored equations. The portion of the inverse is printed to a file with extension `.cii` The sparse form of the matrix only is printed in the form i j C^{ij}, that is, elements of C^{ij} that were not needed in the estimation process are not included in the file.
`!FACPOINTS` *n*	affects the number of distinct points recognised by the `fac()` model function (Table 6.1). The default value of n is 1000 so that points closer than 0.1% of the range are regarded as the same point.

5.8 Job control qualifiers

Table 5.6: List of very rarely used job control qualifiers

qualifier	action
`!KNOTS` *n*	changes the default knot points used when fitting a spline to data with more than *n* different values of the spline variable. When there are more than *n* (default 50) points, ASReml will default to using *n* equally spaced knot points.
`!NOCHECK`	forces ASReml to use any explicitly set spline knot points (see `!SPLINE`) even if they do not appear to adequately cover the data values.
`!NOREORDER`	prevents the automatic reversal of the order of the fixed terms (in the dense equations) and possible reordering of terms in the sparse equations.
`!NOSCRATCH`	forces ASReml to hold the data in memory. ASReml will usually hold the data on a scratch file rather than in memory. In large jobs, the system area where scratch files are held may not be large enough. A Unix system may put this file in the `/tmp` directory which may not have enough space to hold it.
`!POLPOINTS` *n*	affects the number of distinct points recognised by the `pol()` model function (Table 6.1). The default value of *n* is 1000 so that points closer than 0.1% of the range are regarded as the same point.
`!PPOINTS` *n*	influences the number of points used when predicting splines and polynomials. The design matrix generated by the `leg()`, `pol()` and `spl()` functions are modified to include extra rows that are accessed by the `PREDICT` directive. The default value of *n* is 21 if there is no `!PPOINTS` qualifier. The range of the data is divided by *n-1* to give a step size *i*. For each point *p* in the list, a predict point is inserted at $p + i$ if there is no data value in the interval $[p,p{+}1.1{\times}i]$. `!PPOINTS` is ignored if `!PVAL` is specified for the variable. This process also effects the number of levels identified by the `fac()` model term.
`!REPORT`	forces ASReml to attempt to produce the standard output report when there is a failure of the iteration algorithm. Usually no report is produced unless the algorithm has at least produced estimates for the fixed and random effects in the model. Note that residuals are not included in the output forced by this qualifier. This option is primarily intended to help debugging a job that is not converging properly.
`!SCALE 1`	When forming a design matrix for the `spl()` model term, ASReml uses a standardized scale (independent of the actual scale of the variable). The qualifier `!SCALE 1` forces ASReml to use the scale of the variable. The default standardised scale is appropriate in most circumstances.
`!SCORE`	requests ASReml write the SCORE vector and the Average Information matrix to files *basename*`.SCO` and *basename*`.AIM`. The values written are from the last iteration.

Table 5.6: List of very rarely used job control qualifiers

qualifier	action
`!SLOW` n	reduces the update step sizes of the variance parameters more persistently than the `!STEP` r qualifier. If specified, ASReml looks at the potential size of the updates and if any are large, it reduces the size of r. If n is greater than 10 ASReml also modifies the Information matrix by multiplying the diagonal elements by n. This has the effect of further reducing the updates. In the iteration subroutine, if the calculated LogL is more than 1.0 less than the LogL for the previous iteration and `!SLOW` is set and NIT>1, ASReml immediately moves the variance parameters back towards the previous values and restarts the iteration.
`!TOLERANCE` $[s_1\ [\ s_2]]$	modifies the ability of ASReml to detect singularities in the mixed model equations. This is intended for use on the rare occasions when ASReml detects singularities after the first iteration; they are not expected. Normally (when no `!TOLERANCE` qualifier is specified), a singularity is declared if the adjusted sum of squares of a covariable is less than a small constant (η) or less than the uncorrected sum of squares $\times\eta$, where η is 10^{-8} in the first iteration and 10^{-10} thereafter. The qualifier scales η by 10^{s_i} for the the first or subsequent iterations respectively, so that it is more likely an equation will be declared singular. Once a singularity is detected, the corresponding equation is dropped (forced to be zero) in subsequent iterations. If neither argument is supplied, 2 is assumed. If the second argument is omitted, it is given the value of the first. If the problem of later singularities arises because of the low coefficient of variation of a covariable, it would be better to centre and rescale the covariable. If the degrees of freedom are correct in the first iteration, the problem will be with the variance parameters and a different variance model (or variance constraints) is required.
`!VRB`	requests writing of `.vrb` file. Previously, the default was to write the file.

6 Command file: Specifying the terms in the mixed model

6.1 Introduction

The linear mixed model is specified in ASReml as a series of model terms and qualifiers. In this and the following chapter we discuss a functional specification of mixed models in ASReml. This chapter describes the model formula syntax for traditional variance component models.

From Chapter 2, the linear mixed model can be written as

$$\boldsymbol{y} = \boldsymbol{X\tau} + \boldsymbol{Zu} + \boldsymbol{e} \tag{6.1}$$

where $\boldsymbol{y}$ $(n \times 1)$ is a vector of observations, $\boldsymbol{\tau}$ $(p \times 1)$ 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.

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

In this chapter we concentrate on specification of the fixed and random effects and their associated design matrices. For ease of exposition, we assume variance component mixed models (Example 2.2). In these models, the random effects (within model terms) and the residual errors are assumed to be identically and independently distributed (IID). This means they have a common variance and zero covariance. In these variance component models, a functional specification is relatively simple and we discuss this here. In Chapter 7 we present a more general functional specification of random effects and variance structures.

6.2 Specifying model formulae in ASReml

The linear mixed model is specified in ASReml as a series of model terms and qualifiers. Model terms include factor and variate labels (Section 5.4), functions of labels, special terms and interactions of these. The model is specified immediately after the datafile and any job control qualifier and/or tabulate lines. The syntax for specifying the model is

```
NIN Alliance Trial 1989
 variety
⋮
 column 11
nin89.asd !skip 1
yield ∼ mu variety !r idv(repl),
!f mv
residual idv(units)
```

response [*qualifiers*] ∼ *fixed* [`!r` *conrandom*] [`!f` *sparse_fixed*]
[`residual` *conresidual*]

- *response* is the label for the response variable(s) to be analysed; multivariate analysis is discussed in Chapter 8,

- *qualifier*s allow for weighted analysis (Section 6.7) and Generalized Linear Models (Section 6.8),

- ∼ is read as 'modelled as' and separates *response* from the list of fixed and random terms in the linear mixed model,

- *fixed* represents the list of primary fixed explanatory terms, that is, variates, factors, interactions and special terms for which Wald F statistics are required. See Table 6.1 for a brief definition of reserved model terms, operators and commonly used functions. The full definition is in Section 6.6,

- *conrandom* represents the list of consolidated model terms (see Chapter 7) specifying both random effects and variance structures. In this chapter the consolidated model terms are of the form `idv()` with arguments being the explanatory terms to be fitted as random effects, see Table 6.1 and Section 6.6. Specifying `idv(`*term*`)` indicates that the *term* effects are IID distributed with a common variance,

- *sparse_fixed* are additional fixed terms not included in the table of Wald F statistics,

- the `residual` statement allows specification of the residual error variance structure,

- *conresidual* is the list of residual consolidated terms (see Chapter 7) specifying both random effects and variance structures. In this chapter we are assuming that the residual errors are IID. Hence the specification `idv(units)` in the code box, where `units` is the reserved word specifying a factor with a level for every experimental unit.

6.2.1 General rules

The following general rules apply in specifying the linear mixed model

- all elements in the model must be space separated,
- the character $\sim$ ('modelled as') separates the response variables(s) from the explanatory variables in the model,
- elements in the model may be separated by `+` which is ignored except when it is at the end of a line which implies the model continues onto the next line; the `+` sign must appear on the first line of the model statement when the model statement is written over several lines.
- data fields are identified in the model by their labels
  - labels are case sensitive,
  - labels may be abbreviated (truncated) when used in the model line but care must be taken that the truncated form is not ambiguous. If the truncated form matches more than one label, the term associated with the first match is assumed,
    For example, `dens` is an abbeviation for `density` but `spl(dens,7)` is a different model term to `spl(density,7)` because it does not represent a simple truncation.
  - model terms may only appear once in the model line; repeated occurrences are ignored,
  - model terms other than the original data fields are defined the first time they appear on the model line. They may be abbreviated (truncated) if they are referred to again provided no ambiguity is introduced.

    Important It is often clearer if labels are not abbreviated. If abbreviations are used then they need to be chosen to avoid confusion.
- if the model is written over several lines, all but the final line must end with a COMMA (or `+`) to indicate that the list is continued.

In Tables 6.1 and 6.2, the arguments in model term functions are represented by the following symbols

f — the label of a data variable defined as a model factor,

k, *n* — an integer number,

r — a real number,

t — a model term label (includes data variables),

v, *y* — the label of a data variable,

Where a model term takes another model term as an argument, the argument may occasionally need to be predefined. This is done by including the argument model term in the model term list with a leading '-' which will cause the term to be defined but not fitted. For example

```
  Trait.male -Trait.female and(Trait.female)
```

6.2 Specifying model formulae in ASReml

Table 6.1: Summary of reserved words, operators and functions

	model term	brief description	common usage	
			fixed	random
reserved terms	`mu`	the constant term or intercept	√	
	`mv`	a term to estimate missing values	√	
	`Trait`	multivariate counterpart to `mu`	√	
	`units`	forms a factor with a level for each experimental unit		√
operators	`.` or `:`	placed between labels to specify an interaction	√	√
	/	forms nested expansion (Section 6.5)	√	√
	$*$	forms factorial expansion (Section 6.5)	√	√
	`-`	placed before model terms to exclude them from the model	√	√
	,	placed at the end of a line to indicate that the model specification continues on the next line		
	`+`	treated as a space	√	√
	`!{ ... !}`	placed around some model terms when it is important the terms not be reordered (Section 6.4)		√
commonly used functions	`at(`*f*`,`*n*`)`	condition on level *n* of factor *f*. *n* may be a list of level numbers	√	√
	`at(`*f*`)`	forms conditioning covariables for all levels of factor *f*	√	√
	`fac(`*v*`)`	forms a factor from *v* with a level for each unique value in *v*		√
	`fac(`*v*`,`*y*`)`	forms a factor with a level for each combination of values in *v* and *y*		√
	`lin(`*f*`)`	forms a variable from the factor *f* with values equal to 1. . . *n* corresponding to level(1). . . level(*n*) of the factor	√	
	`spl(`*v*`[,`*k*`])`	forms the design matrix for the random component of a cubic spline for variable *v*		√
other functions	*t*{*n*} *t*[*n*]	fits variable *n* from the `!G` set of variables *t*. This is a special case of the `!SUBGROUP` qualifier function applied to `!G` variables. Note that the square parentheses are permitted alternative syntax.	√	√
	`abs(`*v*`)`	forms the absolute value of the variable *v*		
	`and(`*t*`[,`*r*`])`	adds *r* times the design matrix for model term *t* to the previous design matrix; *r* has a default value of 1.predefine it by saying `-t and(t,r)`		√
	`c(`*f*`)`	factor *f* is fitted with *sum to zero* constraints	√	

Table 6.1: Summary of reserved words, operators and functions

model term	brief description	common usage	
		fixed	random
cos(*v*,*r*)	forms cosine from v with period r	✓	
ge(*f*)	condition on factor/variable $f >= r$	✓	
giv(*f*,*n*)	associates the nth .giv G-inverse with the factor f		✓
grm(*f*,*n*)	associates the nth .grm G with the factor f		✓
gt(*f*)	condition on factor/variable $f > r$	✓	
h(*f*)	factor f is fitted *Helmert* constraints	✓	
ide(*f*)	fits pedigree factor f without relationship matrix		✓
inv(*v*[,*r*])	forms reciprocal of $v + r$	✓	
le(*f*)	condition on factor/variable $f <= r$	✓	
leg(*v*,[-]*n*)	forms n+1 Legendre polynomials of order 0 (intercept), 1 (linear)... n from the values in v; the intercept polynomial is omitted if v is preceded by the negative sign.	✓	
lt(*f*)	condition on factor/variable $f < r$	✓	
log(*v*[,*r*])	forms natural logarithm of $v + r$	✓	
ma1(*f*)	constructs MA1 design matrix for factor f		✓
ma1	forms an MA1 design matrix from plot numbers		✓
mbf(*v*,*r*)	is a factor derived from data factor v by using the !MBF qualifier.	✓	✓
out(*n*)	condition on observation n	✓	
out(*n*,*t*)	condition on record n, trait t	✓	
pol(*v*,[-]*n*)	forms n+1 orthogonal polynomials of order 0 (intercept), 1 (linear)... n from the values in v; the intercept polynomial is omitted if n is preceded by the negative sign.	✓	
pow(*x*,*p*[,*o*])	defines the covariable $(x+o)^p$ for use in the model where x is a variable in the data, p is a power and o is an offset.	✓	
qtl(*f*,*p*)	impute a covariable from marker map information at position p	✓	
sin(*v*,*r*)	forms sine from v with period r	✓	
sqrt(*v*[,*r*])	forms square root of $v + r$	✓	
uni(*f*)	forms a factor with a level for each record where factor f is non-zero		✓

Table 6.1: Summary of reserved words, operators and functions

model term	brief description	common usage	
		fixed	random
`uni(`f`,`n`)`	forms a factor with a level for each record where factor f has level n		$\surd$
`vect(`v`)`	is used in a multivariate analysis on a multivariate set of covariates (v) to pair them with the variates	$\surd$	$\surd$

6.2.2 Examples

ASReml code	action

`yield ∼ mu variety`
`residual idv(units)`
	fits a model with a constant and fixed variety effects

`yield ∼ mu variety !r idv(block)`
`residual idv(units)`
	fits a model with a constant term, fixed variety effects and random block effects

`yield ∼ mu time variety time.variety`
`residual idv(units)`
	fits a saturated model with fixed time and variety main effects and time by variety interaction effects

`livewt ∼ mu breed sex breed.sex !r idv(sire)`
`residual idv(units)`
	fits a model with fixed breed, sex and breed by sex interaction effects and random sire effects

6.3 Fixed terms in the model

6.3.1 Primary fixed terms

The *fixed* list in the model formula

- describes the fixed covariates, factors and interactions including special functions to be included in the table of Wald F statistics,
- generally begins with the reserved word `mu` which fits a constant term, mean or intercept, see Table 6.1.

```
NIN Alliance Trial 1989
 variety
:
 row 22
 column 11
nin89.asd !skip 1 !mvinclude
yield ∼ mu variety !r idv(repl),
!f mv
residual idv(units)
```

6.3.2 Sparse fixed terms

The `!f` *sparse_fixed* terms in model formula

- are the fixed covariates (for example, the fixed `lin(row)` covariate now included in the model formula), factors and interactions including special functions and reserved words (for example `mv`, see Table 6.1) for which Wald F statistics are not required,
- include large (>100 levels) terms.

```
NIN Alliance Trial 1989
 variety
⋮
 row 22
 column 11
nin89.asd !skip 1
yield ∼ mu variety !r idv(repl),
!f mv lin(row)
residual idv(units)
```

6.4 Random and residual terms in the variance component model

The `!r` *conrandom* functions have arguments that

- comprise random covariates, factors and interactions including special functions and reserved words, see Table 6.1. Note that `idv()` may not enclose a contracted `at()` function (an `at()` function that is expanded by ASReml to form multiple model terms) because the result is ambiguous.

```
NIN Alliance Trial 1989
 variety
⋮
 row 22
 column 11
nin89.asd !skip 1
yield ∼ mu variety !r idv(repl),
!f mv
residual idv(units)
```

In Chapter 7 we discuss possible qualifiers that allow specification of initial values and constraints. We have given an explicit specification for these variance component models to emphasise the form of the syntax. However, an alternative more concise implicit specification for these models is to note that `idv` is a default function and the random terms can be placed after `!r` without explicitly specifying `idv`. Furthermore, `residual idv(units)` is the default residual specification and may be omitted from the model specification. This is precisely the form used in Release 3 for these models.

6.5 Interactions and conditional factors

6.5.1 Interactions

- interactions are formed by joining two or more terms with a '.' (or a ':' which is replaced with '.'), for example, `a.b` is the interaction of factors `a` and `b`,

- interaction levels are arranged with the levels of the second factor nested within the levels of the first,

- labels of factors including interactions are restricted to 47 characters of which only the first 20 are ever displayed. Thus for interaction terms it is often necessary to shorten the names of the component factors in a systematic way, for example, if `Time` and `Treatment` are defined in this order, the interaction between `Time` and `Treatment` could be specified in the model as `Time.Treat`; remember that the first match is taken so that if the label of each field begins with a different letter, the first letter is sufficient to identify the term,

- interactions can involve model functions.

6.5.2 Expansions

- `+` is ignored, except at the end of the line where it indicates the model is continued on the next line,

- `-` makes sure the following term is defined but does not include it in the model,

- `*` indicates factorial expansion (up to 5 way)
  `a*b` is expanded to `a b a.b`
  `a*b*c*d` is expanded to
  `a b c d a.b a.c a.d b.c b.d c.d a.b.c a.b.d a.c.d b.c.d a.b.c.d`

- `/` indicates nested expansion
  `a/b` is expanded to `a a.b`

- `a.(b c d) e` is expanded to `a.b a.c a.d e`. This syntax is detected by the string '.(' and the closing parenthesis must occur on the same line and before any comma indicating continuation. Any number of terms may be enclosed. Each may have '-' prepended to suppress it from the model.

6.5.3 Conditional factors

A conditional factor is a factor that is present only when another factor has a particular level.

- individual components are specified using the `at(f,n)` function (see Table 6.2), for example, `at(site,1).row` will fit `row` as a factor only for site 1,

- a complete set of conditional terms are specified by omitting the level specification in the `at(`*f*`)` function provided the correct number of levels of *f* is specified in the field definitions,

- otherwise, a list of levels may be specified (see Table 6.2),

- where variable *f* is coded with alphanumeric level names, the level name may be supplied as the second argument. For example `at(Type,TEST).Entry` where `Type` is a factor variable with level names `TEST` and `CONTROL`.
  - `at(`*a*`).b` creates a series of model terms representing `b` nested within `a` for any model term `b`. A model term is created for each level of `a`; each has the size of `b`. For example, if `site` and `geno` are factors with 3 and 10 levels respectively, then `at(site).geno` is shorthand for 3 model terms `at(site,1).geno` `at(site,2).geno` `at(site,3).geno`, each with 10 levels,

  - this is similar to forming an interaction except that a separate model term is created for each level of the first factor; this is useful for random terms when each component can have a different variance. The same effect is achieved by using an interaction (e.g. `site.geno`) and associating a `DIAG` variance structure with the first component (see Section 7.11).

  - any `at()` term to be expanded MUST be the FIRST component of the interaction.
    `geno.at(site)` will not work.
    `at(site,1).at(year).geno` will not work but
    `at(year).at(site,1).geno` is OK.

  - the `at()` factor must be declared with the correct number of levels because the model line is expanded BEFORE the data is read. Thus if site is declared as `site *` or `site !A` in the data definitions,
    `at(site).geno` will expand to
    `at(site,01).geno at(site,02).geno`
    regardless of the actual number of sites.

6.5.4 Associated Factors

Sometimes there is a hierarchical structure to factors which should be recognised as it aids formulation of prediction tables (see `!ASSOCIATE` qualifier on page 187). Common examples are *Genotypes* grouped into *Families* and *Locations* grouped by *Region*. We call these *associated* factors. The key characteristic of associated factors is that they are coded such that the levels of one are uniquely nested in the levels of another. If one is unknown (coded as missing), all associated factors must be unknown for that data record. It is typically unnecessary to interact associated factors except when required to adequately define the variance structure.

6.6 Alphabetic list of model functions

Table 6.2 presents detailed descriptions of the model functions discussed above. Note that some three letter function names may be abbreviated to the first letter.

Table 6.2: Alphabetic list of model functions and descriptions

model function	action
`abs(`v`)`	takes the absolute value of the variable v. This function can be used on the response variable.

`and(`t`,`r`)`
`a(`t`,`r`)`
	overlays (adds) r times the design matrix for model term t to the existing design matrix. Specifically, if the model up to this point has p effects and t has a effects, the a columns of the design matrix for t are multiplied by the scalar r (default value 1.0) and added to the last a of the p columns already defined. The overlaid term must agree in size with the term it overlays. This can be used to force a correlation of *1* between two terms as in a diallel analysis
	`male and(female)`
	assuming the ith male is the same individual as the ith female.

`at(`f`,`n`)`
`@(`f`,`n`)`
	defines a binary variable which is 1 if the factor f has level n for the record. For example, to fit a row factor only for site 3, use the expression `at(site,3).row`. The string `@(` is equivalent to `at(` for this function.

`at(`f`)`
`@(`f`)`
	`at(`f`)` is expanded to a series of terms like `at(`f`,`i`)` where i takes the values `01` to the number of levels of factor f. Since this command is interpreted before the data is read, it is necessary to declare the number of levels of f correctly in its field definition. This extended form may only be used as the first term in an interaction.

`at(`f`,`m`,`n`)` `@(`f`,`m`,`n	`at(`f`,`i`,`j`,`k`)` is expanded to a series of terms `at(`f`,`i`) at(`f`,`j`) at(`f`,`k`)`. Similarly, `at(`f`,`i`).X at(`f`,`j`).X at(`f`,`k`).X` can be written as `at(`f`,`i`,`j`,`k`).`X provided `at(`f`,`i`,`j`,`k`)` is written as the first component of the interaction. Any number of levels may be listed. Contiguous sets of values can be specified as i`:`j.
`cos(`v`,`r`)`	forms cosine from v with period r. Omit r if v is radians. If v is degrees, r is 360.

`con(`f`)`
`c(`f`)`
	apply *sum to zero* constraints to factor f. It is not appropriate for random factors and fixed factors with missing cells. ASReml assumes you specify the correct number of levels for each factor. The formal effect of the `con()` function is to form a model term with the highest level formally equal to minus the sum of the preceding terms. With *sum to zero* constraints, a missing treatment level will generate a singularity but in the first coefficient rather than in the coefficient corresponding to the missing treatment. In this case, the coefficients will not be readily interpretable. When interacting constrained factors, all cells in the cross-tabulation should have data.

`fac(`v`)`
`fac(`v`,`y`)`
	`fac(`v`)` forms a factor with a level for each value of x and any additional points inserted as discussed with the qualifiers `!PPOINTS` and `!PVAL`. `fac(`v`,`y`)` forms a factor with a level for each combination of values from v and y. The values are reported in the `.res` file.

6.6 Alphabetic list of model functions

Table 6.2: Alphabetic list of model functions and descriptions

model function	action

`giv(`*f*`,`*n*`)`
`g(`*f*`,`*n*`)`
`grm(`*f*`,`*n*`)`
	associates the *n*th `.giv` G-inverse with the factor. This is used when there is a known (except for scale) G-structure other than the additive inverse genetic relationship matrix. The G-inverse is supplied in a file whose name has the file extension `.giv` described in Section 9.6. `grm()` and `giv()` are formally equivalent with `grm` standing for generalized relationship Matrix.

`h(`*f*`)`	`h(`*f*`)` requests ASReml to fit the model term for factor *f* using Helmert constraints. Neither Sum-to-zero nor Helmert constraints generate interpretable effects if singularities occur. ASReml runs more efficiently if no constraints are applied. Following is an example of Helmert and sum-to-zero covariables for a factor with 5 levels.

	H1	H2	H3	H4	C1	C2	C3	C4
F1	-1	-1	-1	-1	1	0	0	0
F2	1	-1	-1	-1	0	1	0	0
F3	0	2	-1	-1	0	0	1	0
F4	0	0	3	-1	0	0	0	1
F5	0	0	0	4	-1	-1	-1	-1

model function	action

`ide(`*f*`)`
`i(`*f*`)`
	is used to take a copy of a pedigree factor *f* and fit it without the genetic relationship covariance. This facilitates fitting a *second animal effect.* Thus, to form a direct, maternal genetic and maternal environment model, the maternal environment is defined as a second animal effect coded the same as dams. *viz.* `!r !{ animal dam !} ide(dam)`

`inv(`*v*`[,`*r*`])`	forms the reciprocal of *v* + *r*. This may also be used to transform the response variable.
`leg(`*v*`,[-]`*n*`)`	forms *n*+1 Legendre polynomials of order 0 (intercept), 1 (linear)... *n* from the values in *v*; the intercept polynomial is omitted if *n* is preceded by the negative sign. The actual values of the coefficients are written to the `.res` file. This is similar to the `pol()` function described below.

`lin(`*f*`)`
`l(`*f*`)`
	takes the coding of factor *f* as a covariate. The function is defined for *f* being a simple factor, `Trait` and `units`. The `lin(`*f*`)` function does not centre or scale the variable. Motivation: Sometimes you may wish to fit a covariate as a random factor as well. If the coding is say *1...n*, then you should define the field as a factor in the field definition and use the `lin()` function to include it as a covariate in the model. Do not centre the field in this case. If the covariate values are irregular, you would leave the field as a covariate and use the `fac()` function to derive a factor version.

`log(`*v*`[,`*r*`])`	forms the natural log of *v* + *r*. This may also be used to transform the response variable.

`ma1`
`ma1(`*f*`)`
	creates a first-differenced (by rows) design matrix which, when defining a random effect, is equivalent to fitting a moving average variance structure in one dimension. In the `ma1` form, the first-difference operator is coded across all data points (assuming they are in time/space order). Otherwise the coding is based on the codes in the field indicated.

`mbf(`*f*`,`*c*`)`
`mbf(`*f*`)`
	is a term that is predefined by using the `!MBF` qualifier (see page 71)

6.6 Alphabetic list of model functions

Table 6.2: Alphabetic list of model functions and descriptions

model function	action
mu	is used to fit the intercept/constant term. It is normally present and listed first in the model. It should be present in the model if there are no other fixed factors or if all fixed terms are covariates or contrasts except in the special case of regression through the origin.
mv	is used to estimate missing values in the response variable. Formally this creates a model term with a column for each missing value. Each column contains zeros except for a solitary -1 in the record containing the corresponding missing value. This is used in spatial analyses so that computing advantages arising from a balanced spatial layout can be exploited. The equations for mv and any terms that follow are always included in the sparse set of equations. Missing values are handled in three possible ways during analysis (see Section 6.9). In the simplest case, records containing missing values in the response variable are deleted. For multivariate (including some repeated measures) analysis, records with missing values are not deleted but ASReml drops the missing observation and uses the appropriate unstructured R-inverse matrix. For regular spatial analysis, we prefer to retain separability and therefore estimate the missing value(s) by including the special term mv in the model.

out(n)
out(n,t)
	out(n), out(n,t) establishes a binary variable which is:
	out(i) 1 if data relates to observation i, (trait 1), else is 0
	out(i,t) 1 if data relates to observation i, (trait t), else is 0
	The intention is that this be used to test/remove single observations for example to remove the influence of an outlier or influential point. Possible outliers will be evident in the plot of residuals versus fitted values (see the .res file) and the appropriate record numbers for the out() term are reported in the .res file. Note that i relates to the data analysed and will not be the same as the record number as obtained by counting data lines in the data file if there were missing observations in the data and they have not been estimated. (To drop records based on the record number in the data file, use the !D transformation in association with the !=V0 transformation.)

pol(v,n)
p(v,n)
	forms a set of orthogonal polynomials of order |n| based on the unique values in variate (or factor) v and any additional interpolated points, see !PPOINTS and !PVAL in Table 5.4. It includes the intercept if n is positive, omits it if n is negative. For example, pol(time,2) forms a design matrix with three columns of the orthogonal polynomial of degree 2 from the variable time. Alternatively, pol(time,-2) is a term with two columns having centred and scaled linear coefficients in the first column and centred and scaled quadratic coefficients in the second column.
	The actual values (Robson, 1959, Steep and Torrie, 1960) of the coefficients are written to the .res file. This factor could be interacted with a design factor to fit random regression models. The leg() function differs from the pol() function in the way the quadratic and higher polynomials are calculated.

pow(x,p[,o])	defines the covariable $(x + o)^p$ for use in the model where x is a variable in the data, p is a power and o is an offset. pow(x,0.5[,o]) is equivalent to sqr(x[,o]); pow(x,0[,o]) is equivalent to log(x[,o]); pow(x,-1[,o]) is equivalent to inv(x[,o]).

6.6 Alphabetic list of model functions

Table 6.2: Alphabetic list of model functions and descriptions

model function	action
`qtl(`*f*`,`*r*`)`	calculates an expected marker state from flanking marker information at position *r* of the linkage group *f*(see `!MM` to define marker locations). *r* may be specified as `$TP`*n* where `$TP`*n* has been previously internally defined with a predict statement (see page 183). *r* should be given in Morgans.
`sin(`*v*`,`*r*`)`	forms sine from *v* with period *r*. Omit *r* if *v* is radians. If *v* is degrees, *r* is 360.

`spl(`*v* `[,`*k*`])`
`s(`*v* `[,`*k*`])`
	In order to fit spline models associated with a variate *v* and *k* knot points in ASReml, *v* is included as a covariate in the model and `spl(`*v*`,`*k*`)` as a random term. The knot points can be explicitly specified using the `!SPLINE` qualifier (Table 5.4). If *k* is specified but `!SPLINE` is not specified, equally spaced points are used. If *k* is not specified and there are less than 50 unique data values, they are used as knot points. If there are more than 50 unique points then 50 equally spaced points will be used. The spline design matrix formed is written to the `.res` file. An example of the use of `spl()` is
	`price ∼ mu week !r spl(week)`

`sqrt(`*v*`[,`*r*`])`	forms the square root of $v + r$. This may also be used to transform the response variable.
`Trait`	is used with multivariate data to fit the individual trait means. It is formally equivalent to `mu` but `Trait` is a more natural label for use with multivariate data. It is interacted with other factors to estimate their effects for all traits.
`units`	creates a factor with a level for every record in the data file. This is used to fit the 'nugget' variance when a correlation structure is applied to the residual.
`uni(`*f*`[,0[,`*n*`]])`	creates a factor with a new level whenever there is a level present for the factor *f*. Levels (effects) are not created if the level of factor *f* is `0`, missing or negative. The size may be set in the third argument by setting the second argument to zero.
`uni(`*f*`,`*k*`[,`*n*`])`	creates a factor with a level for every record subject to the factor level of *f* equalling *k*, i.e. a new level is created for the factor whenever a new record is encountered whose integer truncated data value from data field *f* is *k*. Thus `uni(site,2)` would be used to create an independent error term for site 2 in a multi-environment trial and is equivalent to `at(site,2).units`. The default size of this model term is the number of data records. The user may specify a lower number as the third argument. There is little computational penalty from the default but the `.sln` file may be substantially larger than needed. However, if the units vector is full size, the effects are mapped by record number and added back to the fitted residual for creating 'residual' plots.

Table 6.2: Alphabetic list of model functions and descriptions

model function	action

`vect(v)`	is used in a multivariate analysis on a multivariate set of covariates (v) to pair them with the variates. The test example included
	`signal !G 93 # 93 slides`
	`background !G 93`
	`dart.asd !ASUV`
	`signal ∼ Trait Trait.vect(background) ...`
	to fit a slide specific regression of `signal` on `background`. In this example, `signal` is a multivariate set of 93 variates and `background` is a set of 93 covariates. The signal values relate to either the Red or Green channels. So for each slide and channel, we need to fit a simple regression of `signal ∼ mu background` . But the data for the 93 slides is presented in parallel. If it were presented in series, with a factor `slide` indexing the slides, the equivalent model would be `signal ∼ slide slide.background` .

6.7 Weights

Weighted analyses are achieved by using `!WT` *weight* as a qualifier to the response variable. An example of this is `y !WT wt ∼ mu A X` where `y` is the name of the response variable and `wt` is the name of a variate in the data containing weights. If these are relative weights (to be scaled by the `units` variance) then this is all that is required. If they are absolute weights, that is, the reciprocal of known variances, use the `!GF` qualifier to fix the variances in the residual model (Section 7.3). When a structure is present in the residuals (Section 7.3) the weights are applied as a matrix product. If $\boldsymbol{\Sigma}$ is the structure and $\boldsymbol{W}$ is the diagonal matrix constructed from the square root of the values of the variate weight, then $\boldsymbol{R}^{-1} = \boldsymbol{W}\boldsymbol{\Sigma}^{-1}\boldsymbol{W}$. Negative weights are treated as zeros.

6.8 Generalized Linear (Mixed) Models

ASReml includes facilities for fitting the family of Generalized Linear Models (GLMs, McCullagh and Nelder, 1994). A GLM is defined by a mean variance function and a link function. In this context
y is the observation,
n is the count for grouped data specified by the `!TOTAL` qualifier,
ϕ is a parameter set with the `!PHI` qualifier,
μ is the mean on the data scale calculated using the inverse link function from the predicted value η on the underlying scale where $\boldsymbol{\eta} = \boldsymbol{X}\boldsymbol{\tau}$,
v is the variance under some distributional assumption calculated as a function of μ and n, and
d is the deviance (-twice the log likelihood) for that distribution.

Table 6.3: Link qualifiers and functions

Qualifier	Link	Inverse Link	Available with
!IDENTITY	$\boldsymbol{\eta} = \boldsymbol{\mu}$	$\boldsymbol{\mu} = \boldsymbol{\eta}$	All
!SQRT	$\boldsymbol{\eta} = \sqrt{\boldsymbol{\mu}}$	$\boldsymbol{\mu} = \boldsymbol{\eta}^2$	Poisson
!LOGARITHM	$\boldsymbol{\eta} = \ln(\boldsymbol{\mu})$	$\boldsymbol{\mu} = \exp(\boldsymbol{\eta})$	Normal, Poisson, Negative Binomial, Gamma
!INVERSE	$\boldsymbol{\eta} = 1/\boldsymbol{\mu}$	$\boldsymbol{\mu} = 1/\boldsymbol{\eta}$	Normal, Gamma, Negative Binomial
!LOGIT	$\boldsymbol{\eta} = \boldsymbol{\mu}/(1-\boldsymbol{\mu})$	$\boldsymbol{\mu} = \frac{1}{(1+\exp(-\boldsymbol{\eta}))}$	Binomial, Multinomial Threshold
!PROBIT	$\boldsymbol{\eta} = \Phi^{-1}(\boldsymbol{\mu})$	$\boldsymbol{\mu} = \Phi(\boldsymbol{\eta})$	Binomial, Multinomial Threshold
!COMPLOGLOG	$\boldsymbol{\eta} = \ln(-\ln(1-\boldsymbol{\mu}))$	$\boldsymbol{\mu} = 1 - e^{-e^{\boldsymbol{\eta}}}$	Binomial, Multinomial Threshold

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

GLMs are specified by qualifiers after the name of the dependent variable but before the $\sim$ character. Table 6.3 lists the link function qualifiers which relate the linear predictor ($\boldsymbol{\eta}$) scale to the observation ($\boldsymbol{\mu}$ =E[$\boldsymbol{y}$]) scale. Table 6.4 lists the distribution and other qualifiers.

Table 6.4: GLM distribution qualifiers
The default link is listed first followed by permitted alternatives.

<table>
<tr><th>qualifiers</th><th>action</th></tr>
<tr><td colspan="2">!NORMAL [!IDENTITY | !LOGARITHM | !INVERSE]</td></tr>
<tr><td></td><td>allows the model to be fitted on the log/inverse scale but with the residuals on the natural scale. !NORMAL !IDENTITY is the default.</td></tr>
<tr><td colspan="2">!BINOMIAL [!LOGIT | !IDENTITY | !PROBIT | !COMPLOGLOG] [!TOTAL n]</td></tr>
<tr><td>$v = \mu(1-\mu)/n$
$d = 2n(y\ln(y/\mu)$
$+(1-y)\ln(\frac{1-y}{1-\mu}))$</td><td>Proportions or counts [$r = ny$] are indicated if !TOTAL specifies the variate containing the binomial totals. Proportions are assumed if no response value exceeds 1. A binary variate [0, 1] is indicated if !TOTAL is unspecified. The expression for d on the left applies when y is proportions (or binary). The logit is the default link function. The variance on the underlying scale is $\pi^2/3 \sim 3.3$ (underlying logistic distribution) for the logit link.</td></tr>
<tr><td colspan="2">OUTSTANDING: !MULTINOMIAL RT to check</td></tr>
<tr><td colspan="2">!MULTINOMIAL k !CUMULATIVE [!LOGIT | !PROBIT | !COMPLOGLOG] [!TOTAL n]</td></tr>
<tr><td>$v_{ij} = \mu_i(1-\mu_j)/n$
for $i \leq j \leq t$
$d = 2n\Sigma_{i=1}^{k}$
$(y_i\ln(y_i/p_i)$
where
$Y_i = \Sigma_{j=1}^{i} y_j$
μ_i =E(Y_i) and
$p_i = \mu_i - \mu_{i-1}$</td><td>fits a multiple threshold model with $t = k - 1$ thresholds to polytomous ordinal data with k categories assuming a multinomial distribution.
Typically, the response variable is a single variable containing the ordinal score $(1:k)$ or a set of k variables containing counts (r_i) in the k categories. The response may also be a series of t binary variables or a series of t variables containing counts. If t counts are supplied, the total (including the kth category) must be given in another variable indicated by the !TOTAL qualifier: the multinomial model requires a particular variance structure across the multinomial classes. This is formally specified as residual id(units).mthr(Trait).</td></tr>
</table>

Table 6.4: GLM distribution qualifiers

qualifier	action

	The multinomial threshold model is fitted as a cumulative probability model. The proportions ($y_i = r_i/n$) in the ordered categories are summed to form the cumulative proportions (Y_i) which are modelled with logit (`!LOGIT`), probit (`!PROBIT`) or Complementary LogLog (`!CLOG`) link functions. The implicit residual variance on the underlying scale is $\pi^2/3 \sim 3.3$ (underlying logistic distribution) for the logit link, 1 for the probit link. The distribution underlying the Complementary LogLog link is the Gumbel distribution with implicit residual variance on the underlying scale of $\pi^2/6 \sim 1.65$
	For example
	`Lodging !MULTINOMIAL 4 !CUMULATIVE ~ Trait Variety !r block`
	`predict Variety`
	where Lodging is a factor with 4 ordered categories. Predicted values are reported for the cumulative proportions.

`!POISSON` [`!LOGARITHM` | `!IDENTITY` | `!SQRT`]	
$v = \mu$ $d = 2(y\mathtt{ln}(y/\mu)$ $-(y-\mu))$	Natural logarithms are the default link function. ASReml assumes the Poisson variable is not negative.
`!GAMMA` [`!INVERSE` | `!IDENTITY` | `!LOGARITHM`] [`!PHI` ϕ] [`!TOTAL` n]	
$v = \mu^2/(\phi n)$ $d = 2n(-\phi\mathtt{ln}(\frac{\phi y}{\mu})$ $+\frac{\phi y-\mu}{\mu})$	The inverse is the default link function. n is defined with the `!TOTAL` qualifier and would be degrees of freedom in the typical application to mean-squares. The default value of ϕ is 1.
`!NEGBIN` [`!LOGARITHM` | `!IDENTITY` | `!INVERSE`] [`!PHI` ϕ]	
$v = \mu + \mu^2/\phi$ $d = 2((\phi+y)\mathtt{ln}(\frac{\mu+\phi}{y+\phi})$ $+y\mathtt{ln}(\frac{y}{\mu}))$	fits the Negative Binomial distribution. Natural logarithms are the default link function. The default value of ϕ is 1.

General qualifiers	
`!AOD`	requests an Analysis of Deviance table be generated. This is formed by fitting a series of sub models for terms in the DENSE part building up to the full model, and comparing the deviances. An example if its use is `LS !BIN !TOT COUNT !AOD ~ mu SEX GROUP` `!AOD` may not be used in association with `PREDICT`.
`!DISP` [h]	includes an *overdispersion* scaling parameter (h) in the weights. If `!DISP` is specified with no argument, ASReml estimates it as the residual variance of the working variable. Traditionally it is estimated from the deviance residuals, reported by ASReml as `Variance heterogeneity`. An example if its use is `count !POIS !DISP ~ mu group`
`!OFFSET` [o]	is used especially with binomial data to include an offset in the model where o is the number or name of a variable in the data. The offset is only included in binomial and Poisson models (for Normal models just subtract the offset variable from the response variable), for example `count !POIS !OFFSET base !DISP ~ mu group` The offset is included in the model as $\boldsymbol{\eta} = \boldsymbol{X}\tau + o$. The offset will often be something like $\ln(n)$.

Table 6.4: GLM distribution qualifiers

qualifier	action
`!TOTAL [`n`]`	is used especially with binomial and ordinal data where n is the field containing the total counts for each sample. If omitted, count is taken as 1.
Residual qualifiers	control the form of the residuals returned in the `.yht` file. The predicted values returned in the `.yht` file will be on the linear predictor scale if the `!WORK` or `!PVW` qualifiers are used. They will be on the observation scale if the `!DEVIANCE`, `!PEARSON`, `!RESPONSE` or `!PVR` qualifiers are used.
`!DEVIANCE`	produces deviance residuals, the signed square root of d/h from Table 6.4 where h is the dispersion parameter controlled by the `!DISP` qualifier. This is the default.
`!PEARSON`	writes Pearson residuals, $\frac{y-\mu}{\sqrt{v}}$, in the `.yht` file
`!PVR`	writes fitted values on the response scale in the `.yht` file. This is the default.
`!PVW`	writes fitted values on the linear predictor scale in the `.yht` file.
`!RESPONSE`	produces simple residuals, $y - \mu$
`!WORK`	produces residuals on the linear predictor scale, $\frac{y-\mu}{d\mu/d\eta}$

A second dependent variable may be specified (except with a multinomial response (`!MULTINOMIAL`)) if a bivariate analysis is required but it will always be treated as a normal variate (no syntax is provided for specifying GLM attributes for it). The `!ASUV` qualifier is required in this situation for the GLM weights to be utilized.

ASReml internally calculates the appropriate inverse R structure arising from the distribution and link function and so in general a residual line is not needed. The only exception is in this bivariate case when `id(units).us(Trait)` is needed and `us(Trait)` has the three residual components and often the first one associated with the GLM is constrained to an initial value of 1.

6.8.1 Generalized Linear Mixed Models

This section was written by Damian Collins

A Generalized Linear Mixed Model (GLMM) is an extension of a GLM to include random terms in the linear predictor. Inference concerning GLMMs is impeded by the lack of a closed form expression for the likelihood. ASReml currently uses an approximate likelihood technique called penalized quasi-likelihood, or PQL (Breslow and Clayton, 1993), which is based on a first order Taylor series approximation. This technique is also known as Schalls technique (Schall, 1991), pseudo-likelihood (Wolfinger and OConnell, 1993) and joint maximisation (Harville and Mee, 1984, Gilmour *et al.*, 1985). Implementations of PQL are found in many statistical packages, for instance, in the GLMM (Welham, 2005) and the IRREML procedures of Genstat (Keen, 1994), the MLwiN package (Goldstein *et al.*, 1998), the GLMMIX macro in SAS (Wolfinger, 1994), and in the GLMMPQL function in R.

The PQL technique is well-known to suffer from estimation biases for some types of GLMMs. For grouped binary data with small group sizes, estimation biases can be over 50% (e.g. Breslow and Lin, 1995, Goldstein and Rasbash, 1996, Rodriguez and Goldman, 2001, Waddington *et al.*, 1994). For other GLMMs, PQL has been reported to perform adequately (e.g. Breslow, 2003). McCulloch and Searle (2001) also discuss the use of PQL for GLMMs.

The performance of PQL in other respects, such as for hypothesis testing, has received much less attention, and most studies into PQL have examined only relatively simple GLMMs. Anecdotal evidence suggests that this technique may give misleading results in certain situations. Therefore we cannot recommend the use of this technique for general use, and it is included in the current version of ASReml for advanced users. If this technique is used, we recommend the use of cross-validatory assessment, such as applying PQL to simulated data from the same design (Millar and Willis, 1999).

The standard GLM Analysis of Deviance (`!AOD`) should not be used when there are random terms in the model as the variance components are reestimated for each submodel

6.9 Missing values

6.9.1 Missing values in the response

It is sometimes computationally convenient to estimate missing values, for example, in spatial analysis of regular arrays, see example **3a** in Section 7.5. Missing values are estimated if the model term `mv` is included in the model. `mv` is formally shown here in the *sparse fixed effects* to emphasise that it is always included in the sparse equations. If `mv` is listed in the fixed effects section, it and any following fixed effect terms are processed as *sparse* (see Section 6.10.1).

```
NIN Alliance Trial 1989
 variety
⋮
 row 22
 column 11
nin89.asd !skip 1
yield ~ mu variety !r idv(repl),
!f mv
residual idv(units)
```

Formally, `mv` creates a factor with a covariate for each missing value. The covariates are coded 0 except in the record where the particular missing value occurs, where it is coded -1. The action when `mv` is omitted from the model depends on whether a univariate or multivariate analysis is being performed. For a univariate analysis, ASReml discards records which have a missing response. In multivariate analyses, all records are retained and the `R` matrix is modified to reflect the missing value pattern.

6.9.2 Missing values in the explanatory variables

ASReml will abort the analysis if it finds missing values in the design matrix which are not directly associated with missing values for the response or logically excluded from the model by being in combination with an `at()` term which evaluates to ZERO unless `!MVINCLUDE` or `!MVREMOVE` is specified, see Section 5.8. `!MVINCLUDE` causes the missing value to be treated as a zero. `!MVREMOVE` causes ASReml to discard the whole record. Records with missing values in particular fields can be explicitly dropped using the `!DV *` transformation, Table 5.1.

Covariates: Treating missing values as zero in covariates is usually only acceptable if the covariate is centred (has mean of zero).

Design factors: Where the factor level is zero (or missing and the `!MVINCLUDE` qualifier is specified), no level is assigned to the factor for that record. These effectively defines an extra level (class) in the factor which becomes a *reference* level.

6.10 Some technical details about model fitting in ASReml

6.10.1 Sparse versus dense

ASReml partitions the terms in the linear model into two parts: a *dense* set and a *sparse* set. The partition is at the `!r` point unless explicitly set with the `!DENSE` data line qualifier or `mv` is included before `!r`, see Table 5.5. The special term `mv` is always included in sparse. Thus *random* and *sparse* terms are estimated using sparse matrix methods which result in faster processing. The inverse coefficient matrix is fully formed for the terms in the dense set. The inverse coefficient matrix is only partially formed for terms in the sparse set. Typically, the sparse set is large and sparse storage results in savings in memory and computing. A consequence is that the variance matrix for estimates is only available for equations in the dense portion.

6.10.2 Ordering of terms in ASReml

The order in which estimates for the fixed and random effects in linear mixed model are reported will usually differ from the order the model terms are specified. Solutions to the mixed model equations are obtained using the methods outlined Gilmour *et al.*, 1995. ASReml orders the equations in the sparse part to maintain as much sparsity as it can during the solution. After absorbing them, it absorbs the model terms associated with the dense equations in the order specified.

6.10.3 Aliassing and singularities

A singularity is reported in ASReml when the diagonal element of the mixed model equations is effectively zero (see the `!TOLERANCE` qualifier) during absorption. It indicates there is either

- no data for that fixed effect, or
- a linear dependence in the design matrix means there is no information left to estimate

the effect.

ASReml handles singularities by using a generalized inverse in which the singular row/column is zero and the associated fixed effect is zero. Which equations are singular depends on the order the equations are processed. This is controlled by ASReml for the sparse terms but by the user for the dense terms. They should be specified with main effects before interactions so that the table of Wald F statistics has correct marginalization. Since ASReml processes the dense terms from the bottom up, the first level (the last level processed) is typically singular.

The number of singularities is reported in the `.asr` file immediately prior to the REML log-likelihood (`LogL`) line for that iteration (see Section 14.3). The effects (and associated standard or prediction error) which correspond to these singularities are zero in the `.sln` file.

Singularities in the *sparse_fixed* 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. This situation is not easily detected as the only evidence will be in the `.sln` file where different fixed effects are singular. A likelihood ratio test is not valid if the fixed model has changed.

6.10.4 Examples of aliassing

The sequence of models in Table 6.5 are presented to facilitate an understanding of over-parameterised models. It is assumed that `var` is a factor with 4 levels, `trt` with 3 levels and `rep` with 3 levels and that all `var.trt` combinations are present in the data.

Table 6.5: Examples of aliassing in ASReml

model	number of singularities	order of fitting
`yield ~ var !r idv(rep)`	0	`rep var`
`yield ~ mu var !r idv(rep)`	1	`rep mu var` first level of `var` is aliassed and set to zero
`yield ~ var trt !r idv(rep)`	1	`rep var trt` `var` fully fitted, first level of `trt` is aliassed and set to zero

`yield ~ mu var trt var.trt,`
`!r idv(rep)`
	8	`rep mu var trt var.trt`
		first levels of both `var` and `trt` are aliassed and set to zero, together with subsequent interactions

Table 6.5: Examples of aliassing in ASReml

model	number of singularities	order of fitting
`yield ~ mu var trt !r idv(rep), !f var.trt`	8	[`var.trt rep`] `mu var trt` `var.trt` fitted before `mu`, `var` and `trt`, `var.trt` fully fitted; `mu`, `var` and `trt` are completely singular and set to zero. The order within [`var.trt rep`] is determined internally.

6.11 Wald F Statistics

The so called ANOVA table of Wald F statistics has 4 forms:

```
Source   NumDF            F-inc
Source   NumDF            F-inc  F-con M
Source   NumDF  DDF_inc   F-inc           P-inc
Source   NumDF  DDF_con   F-inc  F-con M  P-con
```

depending on whether conditional Wald F statistics are reported (requested by the `!FCON` qualifier) and whether the denominator degrees of freedom are reported. ASReml always reports incremental Wald F statistics (`F-inc`) for the fixed model terms (in the DENSE partition) conditional on the order the terms were nominated in the model. **Note that probability values are only available when the denominator degrees of freedom is calculated**, and this must be explicitly requested with the `!DDF` qualifier in larger jobs. Users should study Section 2.5 to understand the contents of this table. The 'conditional maximum' model used as the basis for the conditional F statistic is spelt out in the `.aov` file described in Section 14.4.

The numerator degrees of freedom (`NumDF`) for each term is easily determined as the number of non-singular equations involved in the term. However, in general, calculation of the denominator degrees of freedom (`DDF`) is not trivial. ASReml will by default attempt the calculation for small analyses, by one of two methods. In larger analyses, users can request the calculation be attempted using the `!DDF` qualifier (page 67). Use `!DDF -1` to prevent the calculation to save processing time when significance testing is not required.

7 Command file: Specifying the variance structures

In Chapter 2 we presented the general linear mixed model

$$\boldsymbol{y} \quad = \quad \boldsymbol{X\tau} + \boldsymbol{Zu} + \boldsymbol{e}$$

where $\boldsymbol{y}$ $(n \times 1)$ is a vector of observations, $\boldsymbol{\tau}$ $(p \times 1)$ 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, see model (2.1). Among the key concepts regarding this model are:

- the *sigma parameterization* (Section 2.1.1):

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

  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 $\boldsymbol{\sigma}_r$. Under this parameterization

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

- G structures for the random model terms (Section 2.1.3) and R structures for the residual error term (Section 2.1.5),

- direct sum structures for $\boldsymbol{G}$ and/or $\boldsymbol{R}_v$ ($\boldsymbol{R}_c$, see below) (Sections 2.1.3 and 2.1.5),

- direct product structures for terms composed of several component factors (Section 2.1.10),

- the *gamma parameterization* for estimation of variance structure parameters as ratios relative to the residual error variance (Section 2.1.6):

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

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

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

In this chapter we give a detailed account of variance modeling in ASReml.

7.1 Applying variance models to random terms

In the previous chapter we showed how to specify the random model terms $\boldsymbol{u}_i$ in $\boldsymbol{u}$ and associated design matrices and we assumed the effects were IID by using an `idv()` function. We can naturally extend this using other functions. Some common variance functions are defined in Table 7.1; the full range of variance model functions and their detailed definition is presented in Table 7.6.

The models are classified as variance models if they include a scale parameter, or as correlation models if their scale is fixed. Except for the `giv` models, correlation models take value 1 on the diagonal. Names of correlation models can be appended with `v` (eg. `idv()`) to add a common variance, ie. same variance across all rows, or with `h` (eg. `idh()`) to allow a separate variance for each row. If all of the variables in a term do not have a variance model specified then the default variance model, `idv()`, will be applied to these variables. We further generalise this in Section 7.2 and Table 7.2 by introducing the idea of a consolidated model term that simultaneously defines both the design matrix ($\boldsymbol{Z}_i$) and variance model ($\boldsymbol{G}_i$), in particular allowing $\boldsymbol{G}_i$ to be the direct product of variance structures. In Section 7.4 we further generalise the consolidated model specification to allow the residual variance structure to be the direct sum of variance structures.

7.2 Process to define a consolidated model term

Consider a linear model term `column.row` comprising the interaction between the single factors column and row. We refer to `column.row` as a *compound model term*. If the variance structure for `column.row` is the direct product of two matrices, the first of which is an IID variance structure, that is, a scaled identity matrix, with dimension equal to the number of levels of the factor `column` and the second of which is a matrix with dimension equal to the number of levels of the factor `row` and with elements representing a first order autoregressive correlation structure `AR1`, then we represent this by the *consolidated model term* `idv(column).ar1(row)`. This specifies a two-dimensional separable spatial variance structure for `column.row` but with spatial correlation in the `row` direction only. A consolidated model term is therefore comprised of component terms, each with a variance model function applied to give the required direct product form of the variance structure. Table 7.2 demonstrates how to build consolidated terms in ASReml for a small selection of examples. The linear model term (single or compound) is first identified (*column 2*) and the individual components that identify the dimension of the individual matrices used in forming the direct product variance structure are then written down (*column 3*). Note that in the simplest cases there is only one component. The variance structure associated with each component has a

Table 7.1: List of common variance model functions, their type (correlation or variance), the form of the variance matrix generated ($\boldsymbol{C}$ for correlation, $\boldsymbol{V}$ for variance matrix, $\boldsymbol{S}$ for scaled variance matrix), and a brief description. Parameters $\sigma_i^2 > 0$ are variances, $-1 < \rho_i < 1$ are correlations. Subscipt c denotes parameter held in common across all rows/columns.

name	type	variance matrix (for set of n effects)	description
`id()`	correlation	$\boldsymbol{C} = \boldsymbol{I}$	IID with variance 1
`idv()`	variance	$\boldsymbol{V} = \sigma_c^2\boldsymbol{I}$	IID with common variance = default model
`idh()`	variance	$\boldsymbol{V} = \text{diag}\{\sigma_1^2 \ldots \sigma_n^2\}$	independent with separate variances
`ar1()`	correlation	$C_{ij} = \rho_c^{\lvert i-j \rvert}$	auto-regressive structure of order 1
`ar1v()`	variance	$V_{ij} = \sigma_c^2\rho_c^{\lvert i-j \rvert}$	auto-regressive structure of order 1
`ar1h()`	variance	$V_{ij} = \sigma_i\sigma_j\rho_c^{\lvert i-j \rvert}$	auto-regressive structure of order 1
`corg()`	correlation	$C_{ij} = \rho_{ij}$	unstructured correlation matrix
`diag()`	variance	$\boldsymbol{V} = \text{diag}\{\sigma_1^2 \ldots \sigma_n^2\}$	independent with separate variances (same as `idh()`)
`grm()`	scaled variance	$\boldsymbol{S}$ specified	applies a known scaled variance matrix; the number of rows in the matrix must be match the number of levels of the factor it is applied to and the order of the rows must match the order of the levels
`nrm()`	scaled variance	$\boldsymbol{S}$ specified	applies a generated relationship matrix derived from the functions argument associated pedigree file
`us()`	variance	$V_{ij} = \sigma_{ij}$	general unstructured, symmetric positive definite covariance matrix
`xfa`k`()`	variance	$\boldsymbol{V} = \boldsymbol{\Lambda}\boldsymbol{\Lambda}^{\mathsf{T}} + \boldsymbol{\Psi}$	factor analytic model of order k with $\boldsymbol{\Lambda}$ of size $n \times k$.

structure name (*column 4*) and a corresponding variance model function name (*column 5*) giving the associated component variance structures (*column 6*). The consolidated model term is the term presented in the final column of the table. In contrast, in ASReml 3 the linear model terms are defined on the model line and subsequently a G structure line is given for each linear model term which specifies the component terms and their associated structures. The simplest form of a consolidated model term is a single model term with a variance model function applied, eg. `idv(repl)` in Table 7.2, and the next simplest is a compound model term with a variance model function applied, eg. `idv(A.B)` in Table 7.2.

In summary, the following are rules in forming consolidated model terms and applying variance model functions to random model terms:

- variance model functions can be applied to single model terms (see example **1** in Table 7.2), the components in compound model terms (examples **4** to **6**) and constructed linear model functions of variables (example **2**),
- variance model functions can also be applied to compound model terms (example **3**)

Table 7.2: Building consolidated model terms in ASReml

	linear model term (*type of term*)	component(s)	variance structure name	variance model function name	covariance component	consolidated model term
1	`repl` *single*	`repl`	IDV	`idv()`	`idv(repl)`	`idv(repl)`
2	`fac(x)` *single*	`fac(x)`	EXPV	`expv()`	`expv(fac(x))`	`expv(fac(x))`
3	`A.B` *compound*	`A.B`	IDV	`idv()`	`idv(A.B)`	`idv(A.B)`

4	`column.row`
	compound
		`column`
		`row`
			IDV
			AR1
				`idv()`
				`ar1()`
					`idv(column)`
					`ar1(row)`
						`idv(column).ar1(row)`

5	`site.variety`
	compound
		`site`
		`variety`
			DIAG
			ID
				`diag()`
				`id()`
					`diag(site)`
					`id(variety)`
						`diag(site).id(variety)`

6	`Trait.animal`
	compound
		`Trait`
		`animal`
			US
			NRM
				`us()`
				`nrm()`
					`us(Trait)`
					`nrm(animal)`
						`us(Trait).nrm(animal)`

- variance model functions *cannot* be applied to expandable model terms, for example, to
  - `A*B` which expands to `A B A.B`
  - `A/B` which expands to `A A.B`
  - `at(A,i,j).B` which expands to `at(A,i).B at(A,j).B`
- only one component of a variance structure for a compound model term may include a variance parameter, the other components must be correlation structures (no associated variance parameter). This is due to the identifiability issues that occur when multiple variance structures are specified. This is explained in NIN example **3a**, see Section 7.5. The defined variance function may be homogeneous (name ending in `v`) or heterogeneous variance (name ending in `h`). This is discussed in detail in Section 7.11.1.

7.2.1 Modelling a single variance structure over several model terms

This facility was motivated by two considerations. Typically the random effects from any two distinct model terms are uncorrelated. However, in some models one $\boldsymbol{G}$ structure may apply across several model terms. Sometimes one also wishes to partition the random effects into sets with independent variance structures. In ASReml, we can accomplish these two models using the special variance model function `str()`, where the name `str` is for *structure* and `str()` has the following general form:

`str(`*model term(s)   variance structure(s)*`)`

The m individual model terms generate the design matrices $\boldsymbol{Z}_i$ and effect vectors $\boldsymbol{u}_i$ of size b_i (i=1,...,m) and the v variance structure terms generate variance structures $\boldsymbol{G}_j$ of size b_j^* ($j = 1, \ldots, v$). The function `str()` generates a combined model design matrix $\boldsymbol{Z}_c = [\boldsymbol{Z}_1 \ldots \boldsymbol{Z}_m]$ and a combined effects vector $\boldsymbol{u}_c^{\mathsf{T}} = [\boldsymbol{u}_1^{\mathsf{T}} \ldots \boldsymbol{u}_m^{\mathsf{T}}]$ of size $b_c = \Sigma_{i=1}^m b_i$ and the variance structure for $\boldsymbol{u}_c$ is $\boldsymbol{G}_c = \oplus_{j=1}^v \boldsymbol{G}_j$ for $\boldsymbol{u}_c$ and $\boldsymbol{G}_c$ to be conformable $\Sigma_{j=1}^v b_j^* = b_c$. If $v = 1$ then there is one variance structure associated with the combined set of effects and if $v > 1$ we can partition $\boldsymbol{u}_c$ and $\boldsymbol{G}_c$ with $\boldsymbol{u}_c^{\mathsf{T}} = [\boldsymbol{u}_1^{*^{\mathsf{T}}} \ldots \boldsymbol{u}_v^{*^{\mathsf{T}}}]$ and $\boldsymbol{G}_c = [\boldsymbol{G}_1^* \ldots \boldsymbol{G}_v^*]$ and the effect vectors are independent of each other and the effects $\boldsymbol{u}_j^*$ have variance structure $\boldsymbol{G}_j^*$. A restriction with `str()` is that the closing parenthesis must be on the same line because of the way ASReml processes the command file.

Example 7.1 Random coefficient regression

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

```
str(Animal age.Animal us(2).id(10))
```

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

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

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

Here, the set of animal intercepts has a common variance (σ_{II}), and the set of animal slopes has a (different) common variance (σ_{SS}). Intercepts and/or slopes from two different animals are independent, but the intercept and slope from any given animal have covariance σ_{IS} (or correlation $\sigma_{IS}/\sqrt{\sigma_{II}\sigma_{SS}}$). In this context, we use integers as arguments to emphasize that the arguments are specifying the size of the variance structure. For this example, `id(10)` can be replaced by `id(Animal)`. In order to simplify processing of the `str()` arguments, ASReml expects at least 1 single term in the consolidated model term to be a variance model function with a dimension rather than a variable name as the argument, eg. `us(2)` in the example. Mostly this is quite natural as a suitable factor is not normally available to indicate the number of linear model terms being combined (2 in this example). The dummy identity function `id(1)` could be introduced to allow processing if the consolidated model term could only be expressed using variable arguments, for example,

```
str(Sire and(Dam) id(1).nrm(Animal))
```

overlays the `Sire` and `Dam` design matrices and allows the resulting effects to be modelled with a single relationship matrix parameter. This random regression model has been developed to describe the form of the `str()` function. We note that this model is equivalent to

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

Example 7.2 Fitting a genetic covariance between direct and maternal effects

This example fits direct effects for two traits, but maternal effects for the first trait only

```
str(Trait.animal at(Trait,1).dam us(3).nrm(animal))
```

where `animal` and `dam` are coded using the pedigree file.

A rather artificial example of using v greater than 1 is when we have 20 levels in a factor `A` and wish to use one variance for the first 8 levels and another for the last 12 levels. Then

```
str(A idv(8) idv(12))
```

will do this.

7.3 Applying variance structures to the residual error term

In Release 4 the residual error term is also defined using a consolidated model term, and it now appears after a `residual` statement that has been introduced to specify the associated variance structure. We give five examples. Firstly, for the default situation of IID residual errors the error model definition line would be

```
residual idv(units)
```

This second example would specify a separable autoregressive spatial model of order 1 (*AR1*×*AR1)* for the observations from a trial arranged in a rectangular array indexed by the data variables `column` and `row`. To apply this variance structure the observations would need to cover the whole grid, but it would not be necessary to pre-order the data file as rows within columns as ASReml uses the information in `column` and `row` to put the observations into the appropriate row within column order:

```
residual ar1v(column).ar1(row)
```

If there were 3 columns and 23 rows in the previous example, then this third example

```
residual ar1v(3).ar1(23)
```

would be an equivalent coding for the *AR1*×*AR1* model using the dimensions of the factors rather than the factor names. In this case the data records *would need to be* sorted in the order rows within columns because ASReml does not have the information needed to reorder the data internally.

The fourth example assumes variance heterogeneity among the data observations, that is, that the three groups comprising observations 1. . . 23, 24. . . 50, 51. . . 70 have unequal variances:

```
residual idv(23) idv(27) idv(20)
```

The fifth and final example is the default residual variance in a multivariate analysis. Specifying `units` as the first component is crucial as ASReml extracts the trait values by trait within unit:

```
residual id(units).us(Trait)
```

7.3.1 Two rules for defining the residual error term

There are two rules in defining the residual that require special consideration:

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 compound model term is specified for the residuals, each combination of levels of the single model terms comprising this term must uniquely identify one unit of the data. For example, in the spatial analysis of a column trial comprising 4 replicates of 24 varieties arranged as a grid of 4 rows by 24 columns (rows are replicates), a first order separable autoregressive spatial variance structure for the residuals can be specified by the consolidated model term `ar1(column).ar1(row)`, where `column` and `row` are the appropriate columns in the data file. However, the number of data units must be the product of the number of levels for `row` and the number of levels for `column`; 96 in this case. If this is not the case, or if more than one unit is associated with some row column combination, ASReml 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 `!COLUMNFACTOR/!ROWFACTOR` data file qualifiers (see Table 5.2) can be used to augment the data by completing the grid to allow an appropriate analysis.

These rules will always be satisfied for a single section of data with IID errors, that is, $\boldsymbol{R}_v = \boldsymbol{R}_{v_1} = \sigma^2 \boldsymbol{I}_n$, see Example 2.2, defined either by default (ie. with no residual 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 multi-environment trial (MET) analysis) or when non-identity variance model functions are used.

7.3.2 Using `sat()` to specify the residual model term for data with sections

Section 2.1.5 described partitioning the data observations into data *sections* to which separate variance structures are applied. There are three data *sections* in the fourth example on page 115. When variance structures are specified using dimensions rather than factor names (`idv(23)` for section 1, `idv(27)` for section 2,. . . in the example), the data must be ordered into sections and the variance structures must be ordered to match the order of the sections in the data file. It is usually more convenient to use a variable in the data file to identify sections within the data. The data will be sorted internally by ASReml (ie. the data file does not need to be ordered in any particular way) and the variance structures for sections can then be specified using the `sat` function, for example

```
residual sat(section).idv(units)
```

for the simple example with 3 data sections, where `section` is a new column in the data file to separate the data into the three sections: units 1. . . 23, 24. . . 50 and 51. . . 70. The `sat` function (shorthand for *section at*) is new with Release 4 and performs several different tasks:

- it tells ASReml that the variance structure for the residual error term is a direct sum structure (see Section 2.1.5) where the different components of the direct sum apply to the different levels of the sectioning variable in the data file

- 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.*

Often *sections* relate to sites (or trials or experiments) in the case where several related trials are analysed together. For example, consider a MET dataset comprising data for three sites. To model the residuals at each site by a separate AR1×AR1 variance structure, we could write

```
residual sat(site).ar1v(column).ar1(row)
```

Alternatively, an AR1×AR1 variance structure for sites 1 and 3, but an IDV×AR1 structure for site 2, could be coded using `sat` either as

```
residual sat(site,1).ar1v(column).ar1(row),
         sat(site,2).idv(column).ar1(row),
         sat(site,3).ar1v(column).ar1(row)
```

or, more succinctly, as

```
residual sat(site,1,3).ar1v(column).ar1(row) sat(site,2).idv(column).ar1(row)
```

For each of these definitions, ASReml will determine the particular levels in `row` and `column` for each site and hence the appropriate sizes of the AR1 matrices.

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

```
residual sat(site,1,3).ar1(row).ar1(column)
```

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

7.4 Identifiability

Once all components of a compound model term have a variance model function applied, ASReml attempts to determine whether the term is identifiable, that is, the terms that can be separately estimated from (are not confounded with) other terms in the model. If the consolidated model term generates a correlation matrix, for example, the consolidated model term for `A.B` is specified as `id(A).ar1(B)`, then it is usually the case that one wishes to fit a model with this correlation structure but to also allow the effects to have a common variance. When a correlation structure is specified for a consolidated term, either for an R or a G structure, ASReml will detect this and add a common scaled variance parameter. Some users might find it simpler and reduce confusion by specifying terms as variance terms directly. For example, `id(A).ar1(B)` should become either `idv(A).ar1(B)` or `id(A).ar1v(B)`; it is arbitrary which variable the common variance is attached to. If more than one variance model function in the consolidated model term includes variance parameters, for example `idv(A).ar1v(B)`, then the parameters will not all be identifiable and so the user must either change `idv(A)` to `id(A)` and leave `ar1v(B)` as it is, or change `ar1v(B)` to `ar1(B)` and leave `idv(A)` as it is.

7.5 A sequence of variance structures for the NIN data

Having outlined the theory and introduced the functional specification, we pause now to consider an example. The following is a series of six variance structures of increasing complexity for the NIN column trial data (see Chapter 3 for an introduction to these data). For each example we present a code box to the right that contains the functional specification and we present a discussion of this code to the left. We present the model specification explicitly to help the user understand the logic. In some cases, experienced users will wish to take advantage of reducing typing and clarity by using default rules. These are discussed in Section 7.10.

1 Randomised complete blocks analysis: blocks fixed

The only random term in a traditional randomised complete block (RCB) analysis of the NIN data is the residual error term $\boldsymbol{e} \sim N(\boldsymbol{0},\ \sigma_e^2 \boldsymbol{I}_{224})$. The model therefore involves just one R structure (IDV) and no G structure. The variance model function name is `idv` and there is just one consolidated model term; `idv(units)`.

```
NIN Alliance Trial 1989
 variety !A
 id
 pid
 raw
 repl 4
⋮
 row 22
 column 11
nin89.asd !skip 1
yield ~ mu variety repl
residual idv(units)
```

2 RCB analysis: blocks random

The random effects RCB model has 2 random terms to indicate that the total variation in the data is comprised of 2 components; a random replicate term $\boldsymbol{u}_r \sim N(\boldsymbol{0},\ \sigma_r^2 \boldsymbol{I}_4)$ and the residual error term, as in example **1**. The `!r` before `repl` tells ASReml that `repl` is a random term. All random terms must be written after `!r` in the model specification line(s). This model involves both the original IDV R structure and an IDV G structure for the random replicate term. There are now now 2 consolidated model terms; `idv(repl)` and `idv(units)`.

```
NIN Alliance Trial 1989
 variety !A
 id
 pid
 raw
 repl 4
⋮
 row 22
 column 11
nin89.asd !skip 1
yield ~ mu variety !r idv(repl)
residual idv(units)
```

3a Two-dimensional spatial model with spatial correlation in one direction

```
NIN Alliance Trial 1989
 variety !A
 id
 pid
 raw
 repl 4
 ⋮
 row 22
 column 11
nin89aug.asd !skip 1
yield ~ mu variety,
!r idv(repl) !f mv
residual idv(column).ar1(row)
```

The NIN trial was actually laid out in as a rectangular array indexed in the data file by `row` and `column`. We can therefore consider fitting a *spatial* model for the residual term where we allow for autocorrelated errors in the row and/or column direction, see Section 7.3. However, there are missing plots in the original data. Before fitting a spatial analysis, we therefore need to fill out the data file to contain records for the missing plots (ASReml can now fill out the data file using `!ROWFACTOR` and `!COLUMNFACTOR`, see Table 5.2). This allows us to define a separable variance structure for the residual error term that is the kronecker product of a structure for rows and a structure for columns. The example in the code box specifies $\boldsymbol{e} \sim N(\mathbf{0},\ \sigma^2_{ec}\boldsymbol{I}_{11} \otimes \boldsymbol{\Sigma}_r(\rho_r))$, that is, a two-dimensional first order separable autoregressive spatial structure for error but with spatial correlation in the row direction only (IDV×AR1): `ar1(row)` models the $\boldsymbol{\Sigma}_r(\rho_r)$ correlation structure for rows and `idv(column)` models the IDV variance structure for columns. The consolidated model term

`idv(column).ar1(row)`

directly mirrors the algebraic form $\text{var}\,(\boldsymbol{e}) = \sigma^2_{ec}\boldsymbol{I}_{11} \otimes \boldsymbol{\Sigma}_r(\rho_r)$.

Important points

- the same residual variance structure could be achieved by specifying `id(column).ar1v(row)` which mirrors the alternate but equivalent algebraic form $\text{var}\,(\boldsymbol{e}) = \boldsymbol{I}_{11} \otimes \sigma^2_{e_r}\boldsymbol{\Sigma}_r(\rho_r)$. It is arbitrary which variable the common variance is attached to: `column` in the code box, `row` in the latter, see Section 7.4 on identifiability.

- if the correlation structure `id(column).ar1(row)` was specified, ASReml would automatically add a common variance to model $\text{var}\,(\boldsymbol{e}) = \sigma^2_e\boldsymbol{I}_{11} \otimes \boldsymbol{\Sigma}_r(\rho_r)$, see Section 7.4.

- `!f mv` is now included in the model specification. This tells ASReml to estimate the missing values. The `!f` before `mv` indicates that the missing values are fixed effects in the sparse set of terms. An equivalent way of specifying this model is
  `yield ~ mu variety mv !r idv(repl)`
  where `mv` is the last fixed effect term and ASReml will include `mv` and succeeding terms in the sparse set.

- ASReml would report an error if the consolidated model term `idv(column).ar1v(row)` was specified: this would correspond to $\text{var}\,(\boldsymbol{e}) = \sigma^2_e\boldsymbol{I}_{11} \otimes \sigma^2_{e_r}\boldsymbol{\Sigma}_r(\rho_r)$ and σ^2_e and $\sigma^2_{e_r}$ are unidentifiable in this case, that is, it is not possible to estimate them separately.

- this is a univariate analysis in which case ASReml automatically uses the gamma parameterization for estimation, see Section 7.6. Consequently, both the sigmas and the gammas are reported. The user can force ASReml to use the sigma parameterization by placing `!SIGMAP` immediately after the independent variable and before ∼ on the model definition line:

  ```
  yield !SIGMAP ~ mu variety mv,
  ```

  `!SIGMAP` is a new qualifier with Release 4, see also Section 7.6. In this case only the sigmas are reported but they appear twice in the output, that is, in both of the columns headed `sigma` in the `.asr` file, see Chapter 11 of the User Guide for detailed information on output formats in ASReml.

3b Two-dimensional separable autoregressive spatial model

This model extends **3a** by specifying a first order autoregressive correlation structure for columns. The R structure in this case is the kronecker product of two autoregressive correlation matrices, that is, $\text{var}\,(\boldsymbol{e}) = \sigma^2_{e_c}\boldsymbol{\Sigma}_c(\rho_c) \otimes \boldsymbol{\Sigma}_r(\rho_r)$, giving an AR1×AR1 model for error. The consolidated model term in this case is `ar1v(column).ar1(row)` and includes `ar1v(column)` to model the $\sigma^2_{e_c}\boldsymbol{\Sigma}_c(\rho_c)$ variance structure for columns.

```
NIN Alliance Trial 1989
 variety !A
 id
 ⋮
 row 22
 column 11
nin89aug.asd !skip 1
yield ~ mu variety,
!r idv(repl) !f mv
residual ar1v(column).ar1(row)
```

Important points

- the same residual variance structure could be achieved by specifying `ar1(column).ar1v(row)` which mirrors the alternate but equivalent algebraic form $\text{var}\,(\boldsymbol{e}) = \boldsymbol{\Sigma}_c(\rho_c) \otimes \sigma^2_{e_r}\boldsymbol{\Sigma}_r(\rho_r)$.

- if the correlation structure `ar1(column).ar1(row)` was specified, ASReml would automatically add a common variance, see Section 7.4.

- ASReml would report an error if the consolidated model term `ar1v(column).ar1v(row)` was specified as this would correspond to $\text{var}\,(\boldsymbol{e}) = \sigma^2_{e_c}\boldsymbol{\Sigma}_c(\rho_c) \otimes \sigma^2_{e_r}\boldsymbol{\Sigma}_r(\rho_r)$ and $\sigma^2_{e_c}$ and $\sigma^2_{e_r}$ are unidentifiable, that is, it is not possible to estimate them separately, see Section 7.4.

3c Two-dimensional separable autoregressive spatial model with measurement error

This model extends **3b** by adding a random `units` term. Thus
$\mathrm{var}\,(\boldsymbol{u}_r) = \sigma^2_r \boldsymbol{I}_4$, $\mathrm{var}\,(\boldsymbol{u}_\eta) = \sigma^2_\eta \boldsymbol{I}_{242}$ and
$\mathrm{var}\,(\boldsymbol{e}) = \sigma^2_{e_c} \boldsymbol{\Sigma}_c(\rho_c) \otimes \boldsymbol{\Sigma}_r(\rho_r)$. The reserved word `units` tells ASReml to construct an additional random term with one level for each experimental unit, so that a second (independent) error term can be fitted. A `units` term is fitted in the model in cases like this where a variance structure is applied to the errors. An IDV variance structure is specified for `units` to model $\sigma^2_\eta\, \boldsymbol{I}_{242}$. The `units` term is sometimes fitted in spatial models for field trial data to allow for a *nugget* effect. The model now has two terms at the plot (experimental unit) level, that is, a correlated structure defined as an R structure and an uncorrelated structure defined in the G structure. There are now three consolidated model terms; `idv(repl)`, `idv(units)` and `ar1v(column).ar1(row)`. This order is reversed in **4**.

```
NIN Alliance Trial 1989
 variety !A
 id
⋮
 row 22
 column 11
nin89aug.asd !skip 1
yield ~ mu variety,
!r idv(repl) idv(units) !f mv
residual ar1v(column).ar1(row)
```

4 Two-dimensional separable autoregressive spatial model defined as a G structure

This model is equivalent to **3c** but with the spatial model defined as a G structure rather than an R structure. The algebraic form is written alternatively, but equivalently, to the form in **3c**, that is
$\mathrm{var}\,(\boldsymbol{u}_r) = \sigma^2_r \boldsymbol{I}_4$,
$\mathrm{var}\,(\boldsymbol{u}_{cr}) = \sigma^2_{cr_c} \boldsymbol{\Sigma}_c(\rho_c) \otimes \boldsymbol{\Sigma}_r(\rho_r)$ and
$\mathrm{var}\,(\boldsymbol{e}) = \sigma^2_e\, I_{242}$.

```
NIN Alliance Trial 1989
 variety !A
 id
⋮
 row 22
 column 11
nin89aug.asd !skip 1
yield ~ mu variety !r idv(repl),
ar1v(column).ar1(row),
!f mv
residual idv(units)
```

Important points

- the same G structure could be achieved by specifying `ar1(column).ar1v(row)`, see similar comment in example **3b**
- if the variance structure `ar1v(column).ar1v(row)` was specified ASReml would report an error, see identical comment in example **3b**
- estimation is based on the gamma parameterization in which case both the estimated sigmas and the estimated gammas are reported. The user can force ASReml to use the sigma parameterization by placing the `!SIGMAP` qualifier immediately after the independent variable and before ~ on the model definition line. In this case only the sigmas would be reported, but they would be reported twice in the output, see Important points under example **3a**.

7.6 Sigma versus gamma parameterization

From Section 2.1.1, the variance matrix of $\boldsymbol{y}$ is

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

see model (2.3). This is referred to as the *sigma parameterization* and the individual variance structure parameters in $\boldsymbol{\sigma}_g$ and $\boldsymbol{\sigma}_r$ are referred to as *sigmas*. For the case when the variance structure for the residual error term is a scaled correlation matrix, that is, $\boldsymbol{R}_v(\boldsymbol{\sigma}_r) = \sigma_e^2\boldsymbol{R}_c(\boldsymbol{\gamma}_r)$, the variance matrix of $\boldsymbol{y}$ can be written alternatively as

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

see (2.8). This is referred to as the *gamma parameterization* and the variance structure parameters in $\boldsymbol{\gamma}_g$ and $\boldsymbol{\gamma}_r$ are referred to as *gammas*, see Section 2.1.6.

7.6.1 Which parameterization does ASReml use for estimation?

By default, ASReml 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. It is possible to over-ride this default as discussed in the following section. ASReml reports both the gammas and the sigmas when the gamma parameterization is used for estimation. For historical reasons, the sigmas are presented twice (two identical columns) when the sigma parameterization is used for estimation.

ASReml uses the sigma parameterization for analyses other than univariate single site analyses, examples including multi-section analyses, multivariate analyses and repeated measures analysis using R structures that are not the default variance model (ie. scaled identity).

7.6.2 Switching from the gamma to the sigma parameterization

ASReml uses the gamma parameterization by default for univariate single section analyses, see above. However, `!SIGMAP` is a new qualifier with Release 4 that enables the user to force ASReml to use the sigma parameterization this case. This is achieved by placing `!SIGMAP` immediately after the independent variable and before ∼ on the model definition line. For example,

```
yield !SIGMAP ∼ mu variety !r idv(repl) !f mv
residual idv(units)
```

would force ASReml to use the sigma parameterization in NIN example **3a,** see Section 7.5.

Table 7.3 gives the variance model specification for each of the six NIN examples (column 3), the individual terms in $\boldsymbol{G}(\boldsymbol{\sigma}_g)$ and $\boldsymbol{R}_v(\boldsymbol{\sigma}_r)$ under the sigma parameterization (column 4), the sigmas that are estimated under this parameterization (column 5), the individual terms in $\boldsymbol{G}(\boldsymbol{\gamma}_g)$ and $\boldsymbol{R}_v(\boldsymbol{\gamma}_r)$ under the gamma parameterization (column 6) and the gammas that are estimated under this parameterization (column 7).

Table 7.3: G structure for the random terms (magenta) and R structure for the residual error term (cyan) under both the sigma and gamma parameterizations, and the corresponding sigma(s)/gamma(s) under each parameterization for the series of NIN data examples

			sigma parameterization		*gamma* parameterization	
no.	definition	variance model specification	$\boldsymbol{G}(\boldsymbol{\sigma}_g)$ $\boldsymbol{R}_v(\boldsymbol{\sigma}_r)$	sigma(s)	$\boldsymbol{G}(\boldsymbol{\gamma}_g)$ $\boldsymbol{R}_c(\boldsymbol{\gamma}_r)$	gamma(s)
1	RCB analysis: blocks fixed	`residual idv(units)`	$\sigma^2_e \boldsymbol{I}_{224}$	σ^2_e	I_{224}	none

2	RCB analysis: blocks random	`!r idv(repl)`
		`residual idv(units)`
			$\sigma^2_r \boldsymbol{I}_4$
			$\sigma^2_e \boldsymbol{I}_{224}$
				σ^2_r
				σ^2_e
					$\gamma_r \boldsymbol{I}_4$
					I_{224}
						γ_r

3a	Two-dimensional spatial model correlation in one direction	`!r idv(repl)`
		`residual idv(column).ar1(row)`
			$\sigma^2_r \boldsymbol{I}_4$
			$\sigma^2_{e_c} \boldsymbol{I}_{11} \otimes \boldsymbol{\Sigma}_r(\rho_r)$
				σ^2_r
				$\sigma^2_{e_c}, \rho_r$
					$\gamma_r \boldsymbol{I}_4$
					$\boldsymbol{I}_{11} \otimes \boldsymbol{\Sigma}_r(\rho_r)$
						γ_r
						ρ_r

3b	Two-dimensional separable autoregressive spatial model	`!r idv(repl)`
		`residual ar1v(column).ar1(row)`
			$\sigma^2_r \boldsymbol{I}_4$
			$\sigma^2_{e_c} \boldsymbol{\Sigma}_c(\rho_c) \otimes \boldsymbol{\Sigma}_r(\rho_r)$
				σ^2_r
				$\sigma^2_{e_c}, \rho_r, \rho_c$
					$\gamma_r \boldsymbol{I}_4$
					$\boldsymbol{\Sigma}_c(\rho_c) \otimes \boldsymbol{\Sigma}_r(\rho_r)$
						γ_r
						ρ_r, ρ_c

3c	Two-dimensional separable autoregressive spatial model with measurement error	`!r idv(repl),`
		`idv(units)`
		`residual ar1v(column).ar1(row)`
			$\sigma^2_r \boldsymbol{I}_4$
			$\sigma^2_\eta \boldsymbol{I}_{224}$
			$\sigma^2_{e_c} \boldsymbol{\Sigma}_c(\rho_c) \otimes \boldsymbol{\Sigma}_r(\rho_r)$
				σ^2_r
				σ^2_η
				$\sigma^2_{e_c}, \rho_r, \rho_c$
					$\gamma_r \boldsymbol{I}_4$
					$\gamma_\eta \boldsymbol{I}_{234}$
					$\boldsymbol{\Sigma}_c(\rho_c) \otimes \boldsymbol{\Sigma}_r(\rho_r)$
						γ_r
						γ_η
						$\rho_r,\ \rho_c$

4	Two-dimensional separable autoregressive spatial model defined as a G structure	`!r idv(repl),`
		`ar1v(column).ar1(row)`
		`residual idv(units)`
			$\sigma^2_r \boldsymbol{I}_4$
			$\sigma^2_{cr_c} \boldsymbol{\Sigma}_c(\rho_c) \otimes \boldsymbol{\Sigma}_r(\rho_r)$
			$\sigma^2_e \boldsymbol{I}_{224}$
				σ^2_r
				$\sigma^2_{cr_c}, \rho_r, \rho_c$
				σ^2_e
					$\gamma_r \boldsymbol{I}_4$
					$\gamma_{cr_c} \boldsymbol{\Sigma}_c(\rho_c) \otimes \boldsymbol{\Sigma}_r(\rho_r)$
					$\boldsymbol{I}_{224}$
						γ_r
						$\gamma_{cr_c}, \rho_r, \rho_c$

7.7 Variance model function qualifiers

A consolidated model term is comprised of one or more covariance components, where a covariance component is a component of the model term to which a variance model function has been applied, see Section 2.1.8 and Table 7.2. All of the covariance components so far have been of the form

vmfname(component)

where *vmfname* is the variance model function name (in `this font` in first column of Table 7.6) and *component* is a component in the model term. Two single covariance components are `idv(repl)` and `ar1(row)`, see Table 7.2.

A general form for a covariance component is

vmfname(component qualifiers)

where *qualifiers* is an optional list of one or more qualifiers to be applied to the variance structure being defined. A simple example of this is the extension of `idv(repl)` to `idv(repl !INIT 0.65)` which specifies an IDV structure of dimension 4 for replicates (NIN example 2) with an initial variance of 0.65 for the variance component associated with replicates under the sigma parameterization, or an initial variance component ratio of 0.65 for the variance component ratio associated with replicates under the gamma parameterization.

Note that a variance structure of a particular dimension, ω say, can been specified directly as

vmfname(ω *qualifiers)*

For example, `idv(3)` defines the IDV variance structure of dimension 3, that is, $\sigma^2\boldsymbol{I}_3$, and `idv(3 !INIT 1.1)` specifies an initial value of 1.1 for the associated variance component under the sigma parameterization or variance ratio under the gamma parameterization. Likewise, `ar1(10)` specifies an autoregressive correlation structure (AR1) of order 10 and `ar1(10 !INIT 0.4)` specifies this same structure with an initial autocorrelation parameter of 0.4. A simple variance component σ^2 would be defined as `idv(1)`. Note that an integer value for the first argument is only valid in variance model functions associated with residual terms and `str()`.

The full list of variance model function qualifiers is in Table 7.4.

7.7.1 Parameter equality constraints `!=`s

Parameters in a variance model can be set to be equal using the `!=`s qualifier (Table 7.4) where s is a string of letters and/or zeros. Positions in the string correspond to the position of the parameters in the list of parameters for the particular variance model:

Table 7.4: Variance model function qualifiers available in ASReml

status	qualifier	description
existing	`!=`s	s is a list of codes that link parameters sharing a common value; details in Section 7.8.2.
New R4	`!COORD` v	provides coordinates for mapping the effects so that a spatial model can be applied to the effects. It is needed when the coordinates are not in the data file, for example `exp(Trait !COORD 1 2.5 3.5 5 8)`, see Section 7.7.2.
existing	`!F`i	is used with the `own()` variance model function, see Section 7.7.3. The argument i is passed to your *own* program.
existing	`!G`s	s is a list of codes `F`, `Z`, `P` or `U`, one for each parameter, specifying whether the parameter is to be `F`ixed at it's initial value, held at `Z`ero (if legal), kept in the `P`arameter space or is `U`nrestricted, see Section 7.7.4
New R4	`!INIT` v	v is the list of initial values for the variance structure parameters. If. initial values can be obtained from the `.msv`, `.rsv` or `.tsv` file, they override these values, see Section 7.7.5
existing	`!SUBSECTION` f	f is a factor in the data that breaks the section into independent subsections, with subsections having common variance parameters, see Section 7.7.6
existing	`!T`s	is used with the `own()` variance model function to set the parameter types, see Sections 7.7.7 and 7.7.3
existing	`!USE` t	t is a compound model term component used elsewhere in the model; allows this variance structure and its parameters to be the same as that used for t, see Section 7.7.8 for an example.

- all parameters with the same letter in the structure are constrained to be equal
- `1-9`, `a-z` and `A-Z` are all unique so that 61 equalities can be specified. `0` and `.` indicate that the corresponding parameter is not related to any other parameter. A colon generates a sequence, that is, `a:e` is the same as `abcde`
- putting `%` as the first character in s makes the interpretation of codes absolute (so that they apply across structures)
- putting `*` as the first character in s indicates that numbers are for repeat counts, `A-Z` are equality codes and are not different from `a-z` giving only 26 equalities. In this case only `.` represents *unrelated to any other parameter*. Thus `!=*.3A2.` is equivalent to `!=.AAA..` or `!=0aaa00` or `!=BAAACD`. Some users might find the contractions appealing, other users find an explicit definition less error-prone.

Examples are presented in Table 7.5. **Important** This syntax is limited in that it cannot apply relationships to simple variance components (random terms that do not have an explicit variance structure) or to implicit residual variance parameters. The `!VCC` syntax is required for these cases.

Table 7.5: Examples of constraining variance parameters in ASReml

ASReml code	action

`!=ABACBADCBA`	constrains all parameters corresponding to `A` to be equal; similarly for `B` and `C`. The fourth parameter symbol `D` is only associated with one parameter and can be replaced by 0 to indicate that it is unconstrained. This sequence applied to an unstructured (US) 4 4 matrix would make it banded, that is
	`A`
	`B A`
	`C B A`
	`D C B A`

`us(site !GP !=0A0AA0,`
`!INIT .3 .1 .4 .1 .1 .3)`
	this example defines a structure for the genotype by site interaction effects in a multi-environment trial with 3 sites, in which the genotypes are independent random effects within sites but are correlated across sites with equal covariance. The initial value for the common covariance is 0.1.

`fa2(site !G4PZ3P4P !=00000000VVVV,`
`!INIT 4*.9 0 3*.1 4*.2).gen`
	a factor analytic model of order 2 for 4 sites, with equal variance across sites, is specified using this code. For the `fa`*k* variance model functions, ASReml orders the parameters as *the loadings followed by the specific variances*. In this example, the first loading in the second factor is constrained to be equal to zero for identifiability. `P` restricts the magnitude of the loadings and the variances to be positive.

`xfa2(site !=VVVV00000000,`
`#contracted form`
`#!=*4V8.`
`!4P4PZ3P,`
`!INIT 4*0.2 4*1.2 0 3*0.3).gen`
	code for a factor analytic model of order 2 for 4 sites, in which the specific variances are all equal. For the `xfa`*k* variance model functions, ASReml orders the parameters as *the specific variances followed by the loadings* (note that this is different to the ordering for the `fa`*k* variance model functions, see previous example). In this example, the first loading in the second factor is constrained to be equal to zero for identifiability.

7.7.2 New R4 Ways to supply distances in one-dimensional metric based models `!COORD` v

Power models rely on the definition of distance for the associated term. Information for determining distances is supplied either implicitly by applying the variance model function to the `fac()` of the coordinate variables, for example

```
expv(fac(X))
```

where X contains the positions, or explicitly with the !COORD qualifier, for example

```
expv(Time !COORD x)
```

where x is a vector of distances which has to be of length the number of levels of Time. For computational reasons it is useful to have the range of x between 5 and 50.

7.7.3 Your own program !Fi

The OWN variance structure is a facility whereby (advanced) users may specify their own variance structure. This facility requires the user to supply a program MYOWNGDG that reads the current set of parameters, forms the $\boldsymbol{G}$ matrix and a full set of derivative matrices, and writes these to disk. Before each iteration, ASReml writes the own parameters to a file, runs MYOWNGDG (it assumes MYOWNGDG forms the $\boldsymbol{G}$ and derivative matrix) and then reads the matrices back in. An example of MYOWNGDG.f90 is distributed with ASReml. It duplicates the AR1 and AR2 variance structures. The following job fits an AR2 structure using this program.

```
Example of using the OWN structure
rep
blcol
blrow
variety 25
yield
barley.asd !skip 1 !OWN MYOWN.EXE
y ~ variety
residual ar1(10).own2(15 !INIT .2 .1 !TRR !F1)
```

The file written by ASReml has extension .own and appears as follows:

```
   15    2    1
0.6025860D+000.1164403D+00
 This file was written by asreml for reading by your MYOWNGDG program
 asreml writes this file, runs your program and then reads
 shfown.gdg
 which it presumes has the following format:
 The first lines should agree with the top of this file
 specifying the order of the matrices         (   15)
  the number of variance structure parameters (    2)
      and a control parameter you can specify (    1).
 These are written in (3I5) format.  They are followed by
 the list of variance parameters written in (6D13.7) format.
 Follow this with     3 matrices written in (6D13.7) format.
 These are to be each of   120 elements being lower triangle
 row-wise of the G matrix and its derivatives with respect
 to the parameters in turn.
```

This file contains details about what is expected in the file written by your program. The filename used has the same basename as the job you are running with extension .own for the file written by ASReml and `.gdg` for the file your program writes. The type of the parameters is set with the `!T` qualifier, see Section 7.7.7, and the control parameter is set using the `!F` qualifier.

- `!F1` applies to the `own` variance model function. With `own`, the argument of `!F` is passed to the `MYOWNGDG` program as an argument the program can access. This is the mechanism that allows several `OWN` models to be fitted in a single run.

- `!T`*s* is used to set the type of the parameters. It is primarily used in conjunction with the `own` variance model function as ASReml knows the type of the parameters in other cases. The valid type codes are given in Section 7.7.7.

7.7.4 Parameter space constraints `!G`*s*

Each parameter has an associated *constraint* code which may be expressed explicitly with the qualifier `!G`*s*, where *s* is the code. The following is a list of the possible constraint codes.

code	constraint type	description
`P`	in the space	`P` is the default in most cases and attempts to keep the parameter in the theoretical parameter space. It is activated when the update of a parameter would take it outside its space. For example, if an update would make a variance negative, the negative value is replaced by a small positive value. Under the `!GP` condition, repeated attempts to make a variance negative are detected and the value is then *fixed* at a small positive value. This is shown in the output in that the parameter will have the code `B` rather than `P` reported in the variance component table.
`U`	unrestricted	`U` does not limit the updates to the parameter. This allows variance parameters to go negative and correlation parameters to exceed ±1. Negative variance components may lead to problems; the mixed model coefficient matrix may become non-positive definite. In this case the sequence of REML log-likelihoods may be erratic and you may need to experiment with starting values.
`F`	fixed	`F` fixes the parameter at its starting value.
`Z`	zero	`Z` mainly applies to factor analytic models where specific variances and/or loadings may be fixed at zero.

For structures with multiple parameters, the form `!G`*`XXXX`* can be used to specify `F`, `P`, `U` or `Z` for the parameters individually. A shorthand notation allows a repeat count before a code letter. Thus `!GPPPPPPPPPPPPPPZPPPZP` could be written as `!G14PZ3PZP`.

For a `US` model, `!GP` makes ASReml attempt to keep the matrix positive definite. After each AI update, it extracts the eigenvalues of the updated matrix. If any are negative or zero, the

AI update is discarded and an EM update is performed. If the highest LogL value relates to a non-positive definite form for the matrix, ASReml may perform hundreds of iterations and never converge. Several forms of EM update are possible (see !EMFLAG) and the PXEM option will converge faster. Note that this option is not available with the nrm or grm functions. Note also, that the EM update is applied to all of the variance parameters in the particular US model and cannot be applied to only a subset of them. EM updates can be slow to converge and an alternative parameterization using a factor analysis may converge faster and give a more parsimonious parameterization. It may be that there is no variance associated with some levels of the matrix, in which case the dimension of the matrix should be reduced.

7.7.5 New R4 Initial values !INIT v

Prior to Release 4 it was necessary to supply initial values for variance structure parameters except for the default IDV variance structure for a random model term, where the default initial variance (ratio) parameter value was 0.1. In Release 4, it is not generally necessary to supply initial values. In this release, ASReml provides starting or initial values for variance structure parameters based on knowledge of the phenotypic variance of the response. Occasionally these initial values are not adequate and more appropriate values will need to be supplied by the user. In this case the user may have good prior information that can be utilized in forming initial values.

There are several ways to provide initial values. The particular choice will depend on how many values and other variance model function qualifiers are to be specified. The initial values can be provided in a number of ways:

- in the variance structure specification, for example

  ```
  ar1(row !INIT 0.35)
  ```

  sets the initial value of the autocorrelation parameter for ar1(row) at 0.35; when this form is used, all of the values required by the structure must be specified

- by modifying the .tsv or .msv file created in a preliminary run (Section 7.9.1)

- by supplying an .rsv file using !CONTINUE, Section 7.9.2.

Important points

- when initial values are supplied using !INIT, there must be the correct number of values and they must be in the appropriate order, for example, for us() the initial values need to be supplied in the order *lower triangle row-wise*

- for the gamma parameterization (Section 7.6), the variance structure parameters will be gammas; in this case the initial values for the gammas that are variance component ratios will be interpreted by ASReml as ratios.

7.7.6 About subsections !SUBSECTION f

The !SUBSECTION qualifier provides an extension to the sat function of Section 7.3.2 for modelling the residual variance. It allows the case of modelling multiple independent sections of correlated observations with a common variance structure and common parameters within sections. The sections can be of different sizes and any homogeneous variance correlation model in Table 7.6 may be used for the variance structure. This gives an R structure of the form

$$\boldsymbol{R}_v \;=\; \oplus_{i=1}^{s} \boldsymbol{R}_{v_i} \quad \text{where} \quad \boldsymbol{R}_{v_i} \;=\; \oplus_{j=1}^{s_i} \boldsymbol{\Sigma}_{ij}(\boldsymbol{\phi}_i)$$

so $\boldsymbol{R}_{v_i}$ may have a direct sum structure with common parameters. Note that, !SUBSECTION is only available when the residual variance function is expressed in terms of one variance function. !SUBSECTION f performs two tasks similar to those described in Section 7.3.2, that is, defining a direct sum structure for the residual vector in a section, with the number of subsections in section i, s_i, given by the number of levels of the factor f, and pruning the levels of the factor defining the variance structure within each section but allowing common variance parameters across sections. The data needs to be sorted in order of the variable f. The following code would specify a common AR1 structure across sections, assumed sorted in to the appropriate order within the section variable, with an initial spatial autocorrelation parameter of 0.5

```
residual ar1(units !INIT 0.5 !SUBSECTION section)
```

If there was data sorted on date within plot then we might use

```
residual exp(date !INIT 0.2 !SUBSECTION plot)
```

to, specify a common EXP structure across plots.

7.7.7 Parameter types !Ts

Each variance parameter also has a *type* which may be set explicitly with the qualifier !Ts, where s is the type code. The following is a list of the possible parameter types and their code. They are usually set internally, are reported in the .tsv file and are used to define the *parameter space*.

type	code	action if !GP is set
variance	V	forced positive
variance ratio	G	forced positive
correlation	R	$-1 < r < 1$
covariance	C	
positive correlation	P	$0 < r < 1$
loading	L	

7.7.8 Equating variance structures !USE *t*

In some plant breeding applications, it can be convenient to define a variance structure as the sum of two simpler terms. For example, given 1000 `entries` representing 50 related `families`, where relationships were derived from markers, the full relationship matrix (inverse) is dense. But it can be well approximated as the sum of a family component and a diagonal entry component. The reformulation gives a sparser (faster) formulation. But now we have two terms to interact with `xfa1(dtrial)` and both must have the same parameters. That is, instead of fitting

```
xfa1(dTrial).grm3(entry)
```

we fit

```
xfa1(dTrial).grm1(family) xfa1(dTrial).grm2(entry)
```

requiring both `xfa1` terms have the same parameters.

If there are only a few parameters, this can be achieved directly as follows:

```
!ASSIGN QP !GPFPFP
!ASSIGN QE !=%ABCDEFGH
!ASSIGN QI !INIT 0.72631 0.000 .242713 0.000 .882465 .846305 .04419 .743393
xfa1(dTrial $QP $QE $QI).grm1(family),
xfa1(dTrial $QP $QE $QI).grm2(entry)
```

However, for a larger term, the number of parameters required may exceed the available letters in the alphabet. In this case `!VCC` can be used:

```
<DATAFILE NAME> !VCC 1
...
xfa1(dTrial $QP $QI).grm1(family),
xfa1(dTrial $QP $QI).grm2(entry)
21 29 !BLOCKSIZE 8 #parameters 21:28 are equal to parameters 29:36 pairwise
```

An ever better option in this case is to use just one structure twice. The following code associates `xfa1(dTrial)` in `xfa1(dTrial).giv2(entry)` with `xfa1(dTrial)` in `xfa1(dTrial).giv1(family)` , that is, both terms point to the one structure definition:

```
xfa1(dTrial $QP $QI).grm1(family),
xfa1(dTrial !USE xfa1(dTrial)).grm2(entry)
```

Table 7.5 gives examples of constraining variance parameters in ASReml.

7.8 Setting relationships among variance structure parameters

7.8.1 Simple relationships among variance structure parameters

It is possible to define simple equality relationships between variance structure parameters using the `!=`s qualifier, see Section 7.8.2 and Table 7.4. More general relationships between variance structure parameters can be defined by placing the `!VCC` c qualifier on the data file definition line. Unlike the case of parameter equality, all parameters can be accessed and the linear relationship is not limited to equality. However, identification of the parameters is not as easy. Each variance structure parameter (γ_i) is allocated a number i internally. These numbers are reported in the `.tsv` file and some are reported in the structure input section of the `.asr` file. These numbers are used to specify which parameters are to be constrained using this method. **Warning** Unfortunately, the parameter numbers usually change if the model is changed.

- `!VCC` c specifies that there are c lines defining parameter relationships,
- If `!VCC` is used a residual line is required and the parameter relationship lines must occur after this residual line,
- each relationship is specified in a separate line of the form

  $i \quad k * v\_k$ simple case

  $i \quad k * v\_k \quad ... \quad p * v\_p$ `!BLOCKSIZE` n general case

  In this specification,
  - i and $k...p$ are the numbers of the specific variance model parameters and v_m, $m = k \cdots p$ are the associated scale coefficients such that $\gamma\_m \times V\_m$ is equal in value to γ_i, for example

    `5 7 * 1` indicates that $\gamma\_\_7 \times 1 = \gamma\_\_5$, ie. parameter 7 is equal to parameter 5

    `5 7 * .1` indicates that parameter 7 is a tenth of parameter 5

  - $*$ indicates the presence of the scale coefficient v_m for the parameter m;
    - if the coefficient is 1 indicating parameter equality, the * 1 can be omitted, for example `5 7` is a simplified coding of the first example
    - if the coefficient is -1
      $i \quad k * -1$ can be simplified to
      $i \quad -k$
      for example, `5 -7` indicates that parameter 7 is has the same magnitude but opposite sign to parameter 5
  - the `!BLOCKSIZE` n qualifier is used when constraints of the same form are required on

blocks of n contiguous parameters, for example,

`21 29 !BLOCKSIZE 8` equates parameters 29 with 21, 30 with 22, ... 36 with 28.

- a variance structure parameter may only be included in one relationship line; to equate several components, put them all in one list on one line

- where the relationship applies among simple model terms (those without an explicit variance structure, for example `units`), the model term name may be given rather than the parameter number.

These examples are summarized in the following table:

ASReml code	action
`5 7 * 1`	parameter 7 equals parameter 5
`5 7`	simple coding for `5 7 * 1`
`5 7 * .1`	parameter 7 is a tenth of parameter 5
`5 -7`	parameter 7 is the negative of parameter 5
`32 34 35 37 38 39`	for a (4 × 4) `US` matrix given by parameters 31 . . . 40, the covariances (parameters 32 . . . 39) are forced to be equal
`21 29 !BLOCKSIZE 8`	equates parameters 29 with 21, 30 with 22, ... 36 with 28.
`units -uni(check)`	parameter associated with model term `uni(check)` has the same magnitude but opposite sign to the parameter associated with model term `units`.

7.8.2 Fitting linear relationships among variance structure parameters

The user may wish to define relationships between particular variance parameters. For example, consider an experiment in which two or more separate trials are sown adjacent to one another at the same trial site, with trials sharing a common plot boundary. In this case it might be sensible to fit the same spatial parameters and error variances for each trial. In other situations it can be sensible to define the same variance structure over several model terms. ASReml 3 catered for equality and multiplicative relationships among variance parameters. In ASReml 4 linear relationships among variance structure parameters can be defined through a simple linear model and by supplying a design matrix for a set of parameters. The design matrix is supplied as an `ascii` file containing a row for each parameter in a set of contiguous parameters and a column for each *new* parameter. This design matrix is associated with the job through a statement after the residual model definition line(s), of the form:

`VCM` *parameter_number_list new filename*

where *parameter_number_list* is a list of parameters in the set, and can be abbreviated to *first* and *last* if all the intermediate parameters are in the set, *new* is the number of new parameters and *filename* is the name of the file containing the design matrix.

For example, the Wolfinger rats example involves modelling a 5×5 symmetric residual matrix.

```
Wolfinger Rat data
 treat !A
 wt0 wt1 wt2 wt3 wt4
 subject * !=V0
wolfrat.dat !skip 1 !ASUV !VCC 2
wt0 wt1 wt2 wt3 wt4   Trait treat Trait.treat
1 2 0
27 0 ID #error variance
Trait 0 US * #* indicates generates initial values
#uses 15 parameters numbered 5-19 generating symmetric matrix
#5
#6 7
#8 9 10
#11 12 13 14
#15 16 17 18 19
```

Wolfinger (1996) reports the fitting of the HuynhFeldt variance structure to this data. This structure is of the form

$$\begin{aligned}\boldsymbol{\sigma}_{ii} &= \boldsymbol{\sigma}_{ni}\\ \boldsymbol{\sigma}_{ij} &= 1/2\left(\boldsymbol{\sigma}_{ni}+\boldsymbol{\sigma}_{nj}\right)-\boldsymbol{\sigma}_{no} \quad j < i \leq p\end{aligned}$$

In the rats example, the relationship between the original and new parameters is $\boldsymbol{\sigma} = \boldsymbol{M}\boldsymbol{\sigma}_n$ where $\boldsymbol{\sigma}$ and $\boldsymbol{\sigma}_n$ are 15×1 and 6×1 vectors respectively, and $\boldsymbol{M}$ is a 15×6 matrix:

```
1    0    0    0   0     0
0.5  0.5  0    0   0    -1
0    1    0    0   0     0
0.5  0    0.5  0   0    -1
0    0.5  0.5  0   0    -1
0    0    1    0   0     0
0.5  0    0    0.5 0    -1
0    0.5  0    0.5 0    -1
0    0    0.5  0.5 0    -1
0    0    0    1   0     0
0.5  0    0    0   0.5  -1
0    0.5  0    0   0.5  -1
0    0    0.5  0   0.5  -1
0    0    0    0.5 0.5  -1
0    0    0      0   1   0
```

A way of fitting this model would be to put the matrix values in a file `HuynhFeldt.vcm` and use

```
VCM 5 19 6 HuynhFeldt.vcm #parameters 5 to 19 explained in terms of 6 parameters
```

Note that if the user fits another model with differing numbers of variance structure parameters so that the variance structure parameters are renumbered, then all the user needs to do to continue with the same relationships is to change the *parameter_number_list* parameters on the VCM line.

Important The VCM statement must be placed after any residual definition line(s).

The new qualifier !DESIGN on the datafile line causes ASReml to write the design matrix, not including the response variable, to a .des file. It allows ASReml to create the design matrix required by the VCM process, see Section 7.8.2 above. For example, using a control file vcmdes.as containing

```
Create VCM Design for H-F model
Row *
Col *
Off
Y !=V0
vcmdes.asd !DESIGN
Y ~ Row and(Row,-0.5) and(Col,0.5) Off
```

and a data file vcmdes.asd containing

```
1 1  0
2 1 -1
2 2  0
3 1 -1
3 2 -1
3 3  0
4 1 -1
4 2 -1
4 3 -1
4 4  0
5 1 -1
5 2 -1
5 3 -1
5 4 -1
5 5  0
```

then the file vcmdes.des will be generated which contains the values used in fitting the variance model for the HuynhFeldt model given in Section 7.8.2.

7.9 Ways to present initial values to ASReml

In complex models, the Average Information algorithm can have difficulty maximising the REML log-likelihood when starting values are not reasonably close to the REML solution.

ASReml has several internal strategies to cope with this problem. When the user needs to provide better starting values than those generated by ASReml three of the methods are:

- inserting explicit initial values in the `.as` file (for example using `!INIT`),
- doing a preliminary run to obtain `.tsv` or `.msv` files and then modifying the parametric information in one of those files, Section 7.9.1,
- fitting a simpler model and using parameter values derived from the simpler model, through the `.rsv` file, Section 7.9.2.

7.9.1 Using templates to set parametric information associated with variance structures using `.tsv` and `.msv` files

ASReml 3 needed initial values for most variance structure parameters and allowed specification of parametric constraints and relationships (equality and scale) between parameters to be defined. This parametric information was interspersed within the structure definition. Release 4 allows an alternative way of specifying this parametric information, essentially constructing a table in a `.tsv` file, with the rows labelled by the specific parameters, columns for initial values and parametric constraints, and two columns that allow specification of relationships. This `.tsv` file is written by ASReml after the input file has been parsed; using `*` to represent initial values and setting `!MAXITER 0` gives an easy construction. Once the `.tsv` file has been edited it can be read by inserting `!TSV` on the data file line. As an example

```
Wolfinger Rat data
 treat !A
 wt0 wt1 wt2 wt3 wt4
 subject * !=V0
wolfrat.dat !skip 1 !ASUV !MAXITER 0
wt0 wt1 wt2 wt3 wt4   Trait treat Trait.treat
1 2 0
27 0 ID #error variance
Trait 0 US * #* indicates generates initial values
```

generates a `.tsv` file.

```
# This .tsv file is a mechanism for resetting initial parameter values
# by changing the values here and rerunning the job with !TSV
# You may only change values in the last 4 fields.
# Fields are:
# GN, Term, Type, PSpace, Initial_value, RP_GN, RP_scale.

5, "units.us(Trait);us(Trait)_1", G, P, 4.7911110 , 5, 1
6, "units.us(Trait);us(Trait)_2", G, P, 5.0231481 , 6, 1
7, "units.us(Trait);us(Trait)_3", G, P, 15.298889 , 7, 1
8, "units.us(Trait);us(Trait)_4", G, P, 4.8438271 , 8, 1
9, "units.us(Trait);us(Trait)_5", G, P, 11.264815 , 9, 1
```

```
10, "units.us(Trait);us(Trait)_6", G, P, 26.095692 , 10, 1
11, "units.us(Trait);us(Trait)_7", G, P, 4.6882715 , 11, 1
12, "units.us(Trait);us(Trait)_8", G, P, 10.824074 , 12, 1
13, "units.us(Trait);us(Trait)_9", G, P, 27.332887 , 13, 1
14, "units.us(Trait);us(Trait)_10", G, P, 71.875403 , 14, 1
15, "units.us(Trait);us(Trait)_11", G, P, 3.9083333 , 15, 1
16, "units.us(Trait);us(Trait)_12", G, P, 10.292592 , 16, 1
17, "units.us(Trait);us(Trait)_13", G, P, 34.137962 , 17, 1
18, "units.us(Trait);us(Trait)_14", G, P, 69.287036 , 18, 1
19, "units.us(Trait);us(Trait)_15", G, P, 141.97296 , 19, 1
```

Parameter constraints and initial values can be changed by editing the values in the `PSpace` and `Initial_value` columns. Scale relationships can be introduced by noting that the full set of parameters can be related to a subset of parameters and scale factors such as

parameter = *subset parameter* * *scale*

or

`GN` *column parameter*, `RP_GN` *column parameter* * `RP_scale` *value*

where `GN`, `RP_GN` and `RP_scale` are columns in the `.tsv` file. The relationships generated by

```
VCC 2
5 6 8 11 15 7 * 2 9 * 2 12 * 2 16 * 2 #parameters 6 8 11 15 are equal to 5
#7 9 12 16 are twice 5
10 13 17 #parameters 13 and 17 are equal to 10
#the full set of parameters 5-19 can therefore be expressed in terms of the subset parameters
5, 10 ,14, 18 and 19
```

can be introduced by editing the `RN_GN` and `RP_scale` columns. Some users would prefer to insert initial values into this `.tsv` file under the `Initial_value` column. As an example, the file below contains values based on using 4.8, 26, 70, 35 and 70 for parameters 5, 10, 14, 18 and 19. The data values in the `.tsv` file become

```
# GN, Term, Type, PSpace, Initial_value, RP_GN, RP_scale.
5, "units.us(Trait);us(Trait)_1", G, P, 4.8 , 5, 1.0000
6, "units.us(Trait);us(Trait)_2", G, P, 4.8 , 5, 1.0000
7, "units.us(Trait);us(Trait)_3", G, P, 9.6 , 5, 2.0000
8, "units.us(Trait);us(Trait)_4", G, P, 4.8 , 5, 1.0000
9, "units.us(Trait);us(Trait)_5", G, P, 9.6 , 5, 2.0000
10, "units.us(Trait);us(Trait)_6", G, P, 26 ,10, 1.0000
11, "units.us(Trait);us(Trait)_7", G, P, 4.8 , 5, 1.0000
12, "units.us(Trait);us(Trait)_8", G, P, 9.6 , 5, 2.0000
13, "units.us(Trait);us(Trait)_9", G, P, 26 , 10, 1.0000
14, "units.us(Trait);us(Trait)_10", G, P, 70 , 14, 1.0000
15, "units.us(Trait);us(Trait)_11", G, P, 4.8 , 5, 1.0000
16, "units.us(Trait);us(Trait)_12", G, P, 9.6 , 5, 2.0000
17, "units.us(Trait);us(Trait)_13", G, P, 26 , 10, 1.0000
18, "units.us(Trait);us(Trait)_14", G, P, 35 , 18, 1.0000
```

```
19, "units.us(Trait);us(Trait)_15", G, P, 70 , 19, 1.0000
```

Sometimes users wish to rerun a job making changes to the final values, parametric constraints and relationships (equality and scale) between parameters. A file `.msv` is produced, similar to `.tsv` but containing final values that can be edited and used with `!MSV`. If `!TSV` (or `!MSV`) is specified `ASReml` will read the current (created with the same `PART` number) `.tsv` (or `.msv`) file. If there is no current `.tsv` (or `.msv` file), a non-current (produced from a different `PART` of the same job) `.tsv` (or `.msv`) file will be read.

Alternative ways of specifying `!TSV` and `!MSV` are `!CONTINUE 2` and `!CONTINUE 3` and these qualifiers can be used as options on the command line as `-C2` and `-C3`. Note that the constraints in the `.tsv`/`.msv` files take precedence over those in the `.as` file.

7.9.2 Using estimates from simpler models

Sometimes we have estimates from simpler models and we wish to reduce the need for the user to type in updated starting values. The `!CONTINUE` command line qualifier instructs ASReml to update initial parameter values from a `.rsv` file. When it is specified, `ASReml` first looks for a current `.rsv` file, and if found will read it and report the constructed initial values in the `.tsv` file. If there is no current `.rsv` file, it looks for the most recent noncurrent `.rsv` file and uses that to construct initial values. As discussed below, 'current' means having the same 'basename' and 'run number'. A non-current file will have the same 'basename' but a different 'run number'. When reading the `.rsv` file, if the variance structure for a term has changed, ASReml will take results from some structures as supplying starting values for other structures. The transitions recognised are

`CORUH` to `FA1` and `XFA1`
`CORGH` to `US`
`DIAG` to `CORUH`
`DIAG` to `FA1`
`DIAG` to `XFA1`
`FA`*i* to `CORGH`
`FA`*i* to `FA`*i+1*
`FA`*i* to `US`
`XFA`*i* to `XFA`*i+1*
`XFA`*i* to `US`
`US` to `XFA1`, `XFA2`, `XFA3`

Users may wish to keep output from a series of runs. This can be done by using `!RENAME 1 !ARG` *runnumber* on the first line of the command file or alternatively `-R1` *basename runnumber* on the command line. This ensures that the output from the various parts has runnumber appended to the base filename. If an `.rsv` file does not exist for the particular runnumber you are running, `ASReml` will retrieve starting values from the most recent `.rsv` file formed by that job. You can, of course, copy an `.rsv` file building the new *runnumber* into its name so that `ASReml` uses that particular set of values. The `.asr` file keeps track of which `.rsv` files have been formed. If the user wishes to use different models with different runs then using `!DOPART $1` and specifying the different models in different parts will achieve this aim.

7.10 Default variance structures in ASReml

There are default variance structures in ASReml that allow the linear mixed model to be specified more succinctly. IDV is the default variance structure for random model terms and for the residual error terms. For example

- `A` will be interpreted as `idv(A)`
- `A.B` will be interpreted as `idv(A.B)`
- `A.B.C` will be interpreted as `idv(A.B.C)`
- `sat(Expt,1).A` will be interpreted as `sat(Expt,1).idv(A)`
- `sat(Expt,1).A.B` will be interpreted as `sat(Expt,1).idv(A.B)`
- `sat(Expt,1).A.B.C` will be interpreted as `sat(Expt,1).idv(A.B.C)`

In these cases the model term can be followed by an initial value and/or a parametric qualifier, for example

- `A 1 !GP` is interpreted as `idv(A !INIT 1 !GP)`

There is always a residual error term in the model but if it is not explicitly specified it is assumed to be `idv(units)` for univariate data and `id(units).us(Trait)` for multivariate data. If the consolidated model term definition is incomplete, that is, if some but not all of the components have a variance model function specified, the variance model functions `idv()` or `id()` will be applied to these components depending on the variance model functions specified. For example

- `idv(A).B` will be interpreted as `idv(A).id(B)`
- `id(A).B` will be interpreted as `id(A).idv(B`
- `id(A).B.C` will be interpreted as `id(A).idv(B.C)`
- `idv(A).B.C` will be interpreted as `idv(A).id(B.C)`

Similarly, at the residual level as `sat()` cannot be converted into a variance function

- `sat(Expt,1).id(A).B` will be interpreted as `sat(Expt,1).id(A).idv(B)`
- `sat(Expt,1).id(A).B.C` will be interpreted as `sat(Expt,1).id(A).idv(B.C)`
- `sat(Expt,1).idv(A).B.C` will be interpreted as `sat(Expt,1).idv(A).id(B.C)`

However, it is good practice to specify variance model functions for the components in model terms and we encourage the user to do this. ASReml will automatically add a common variance to consolidated model terms that are specified as correlation models for both R and G structures, for example,

- `id(A)` will be converted to `idv(A)`
- `sat(Expt,1).id(units)` will be converted to `sat(Expt,1).idv(units)`
- `id(A).ar1(B)` will be converted to `idv(A).ar1(B)`
- `ar1(A).ar1(B)` will be converted to `ar1v(A).ar1(B)`
- `sat(Expt,1).id(A).ar1(B)` will be converted to `sat(Expt,1).idv(A).ar1(B)`
- `sat(Expt,1).ar1(A).ar1(B)` will be converted to `sat(Expt,1).ar1v(A).ar1(B)`

Using NIN example **2** for demonstration (Section 7.5), a more succinct coding of the model definition would be

```
yield ∼ mu variety !r repl
residual units
```

which would result in identical output to the original example. The model could be relaxed further to

```
yield ∼ mu variety !r repl
```

7.11 Variance model functions available in ASReml

The full range of variance models, that is, correlation, homogeneous variance and heterogeneous variance models available in ASReml is presented in Table 7.6 which is located at the end of this chapter for easy access, see Section 7.12 on page 147. This presents the variance structure name (in UPPERCASE), the corresponding variance model function name (in `lowercase`) used to associate the variance structure with the appropriate component of a model term, a brief description, the algebraic form of the model and the number of associated variance structure parameters.

The models span correlation (base) models (diagonal elements equal to 1 and correlations on the off diagonals), the extension of these to variance models (variances on the diagonals and covariance on the off diagonals), additional models that are parameterized as variance matrices rather than as correlation matrices and some special cases where the covariance structure is known except for the scale.

See Sections 7.2 and 7.10 for important points to note in defining variance structures in ASReml.

7.11.1 Forming variance models from correlation models

The variance function models presented under correlation models in Table 7.6 (`id` . . . `mat`k) are used to specify the correlation models for the corresponding variance structures. The corresponding homogeneous and heterogeneous variance models are specified by appending `v` and `h` to the variance model function names respectively, and appending the corresponding variance parameters to the corresponding list of parameters. This convention holds for most models. It does not make sense to append `v` or `h` to the variance model function names for the heterogeneous variance models from `diag` . . . `xfa`k.

In summary:

- to specify a correlation model, provide the variance model function name given in Table 7.6, for example, for a factor `row`

  `exp(row)`

  is an exponential correlation model with a single correlation parameter,

- to specify an homogeneous variance model, append a `v` to the variance model function name, for example

  `expv(row)`

  is an exponential variance model with 2 parameters (correlation and variance),

- to specify a heterogeneous variance model, append an `h` to the variance model function name, for example

  `coruh(site)`

  is a variance matrix with different variances for each site but the same correlation for all pairs of sites.

Important See Section 7.4 for rules on combining variance models and Section 7.7.5 for important notes regarding initial values.

7.11.2 Non singular variance matrices

For REML estimation, ASReml needs to invert each variance matrix. For this it requires that the matrices be negative definite or positive definite. They must not be singular. Negative definite matrices will have negative elements on the diagonal of the matrix and/or its inverse. There are two exceptions: the `XFA` model which has been specifically designed to fit singular matrices (Thompson *et al.* 2003, page 144), and singular relationship matrices described in Chapter 9.

If an estimated matrix comes too close to being singular, ASReml will stop iterating.

Let $\boldsymbol{x}^\mathsf{T}\boldsymbol{A}\boldsymbol{x}$ represent an arbitrary quadratic form for $\boldsymbol{x} = (x_1, \ldots, x_n)^\mathsf{T}$. The quadratic form is said to be nonnegative definite if $\boldsymbol{x}^{\mathsf{T}}\boldsymbol{A}\boldsymbol{x} \geq \mathbf{0}$ for all $\boldsymbol{x} \in \boldsymbol{R}^n$. If $\boldsymbol{x}^{\mathsf{T}}\boldsymbol{A}\boldsymbol{x}$ is nonnegative definite and in addition the null vector $\mathbf{0}$ is the only value of $\boldsymbol{x}$ for which $\boldsymbol{x}^{\mathsf{T}}\boldsymbol{A}\boldsymbol{x} = \mathbf{0}$, then the quadratic form is said to be positive definite. Hence the matrix $\boldsymbol{A}$ is said to be positive definite if $\boldsymbol{x}^{\mathsf{T}}\boldsymbol{A}\boldsymbol{x}$ is positive definite, see Harville (1997), pp 211.

7.11.3 Notes on the variance models

These notes provide additional information on the variance models defined in Table 7.6.

- the IDH and DIAG models fit the same diagonal variance structure,

- the CORGH and US are equivalent variance structures parameterised differently. Both may fail to converge if the starting values are not good and/or if the maximum *REML* likelihood occurs at parameter values outside the parameter space. The us model is likely to be better when the matrix is of order 3 or higher.

- in CHOLk models $\boldsymbol{\Sigma} = \boldsymbol{L}\boldsymbol{D}\boldsymbol{L}^{\mathsf{T}}$ where $\boldsymbol{L}$ is lower triangular with ones on the diagonal, $\boldsymbol{D}$ is diagonal and k is the number of non-zero off diagonals in $\boldsymbol{L}$,

- in CHOL*k*C models $\boldsymbol{\Sigma} = \boldsymbol{L}\boldsymbol{D}\boldsymbol{L}^{\mathsf{T}}$ where $\boldsymbol{L}$ is lower triangular with ones on the diagonal, $\boldsymbol{D}$ is diagonal and k is the number of non-zero sub diagonal columns in $\boldsymbol{L}$. This is somewhat similar to the factor analytic model.

- in ANTEk models $\boldsymbol{\Sigma}^{-1} = \boldsymbol{U}\boldsymbol{D}\boldsymbol{U}^{\mathsf{T}}$ where $\boldsymbol{U}$ is upper triangular with ones on the diagonal, $\boldsymbol{D}$ is diagonal and k is the number of non-zero off diagonals in $\boldsymbol{U}$,

- the CHOLk and ANTE*k* models are equivalent to the US structure, that is, the full variance structure, when k is $\omega - 1$,

- initial values for US, CHOL and ANTE structures are given in the form of a US matrix which is specified lower triangle row-wise, viz
$$\begin{bmatrix} \sigma_{11} & & \\ \sigma_{21} & \sigma_{22} & \\ \sigma_{31} & \sigma_{32} & \sigma_{33} \end{bmatrix},$$
that is, initial values are given in the order, $1 = \sigma_{11}$, $2 = \sigma_{21}$, $3 = \sigma_{22}, \ldots$

- the US model is associated with several special features of ASReml. There is an process to update its values by EM (see !EMFLAG rather than AI when its AI updates make the matrix non positive definite. Also, when used in the $\boldsymbol{R}$ structure for multivariate data, ASReml automatically recognises patterns of missing values in the responses (see Chapter 8).

7.11.4 Notes on Matérn

The Matérn class of isotropic covariance models is now described. ASReml uses an extended Matérn class which accomodates geometric anisotropy and a choice of metrics for random fields observed in two dimensions. This extension, described in detail in Haskard (2006), is given by

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

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

$$\rho_M(d;\phi,\nu) = \left\{2^{\nu-1}\Gamma(\nu)\right\}^{-1}\left(\frac{d}{\phi}\right)^{\nu} K_\nu\left(\frac{d}{\phi}\right), \tag{7.1}$$

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

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 uses numerical derivatives for ν when its current value is outside the interval [0.2,5].

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

When $\nu = 1.5$ then

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

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

The final metric parameter λ is not estimated by ASReml; it has default value of 2 for Euclidean distance. Setting $\lambda = 1$ provides the cityblock metric, which together with $\nu = 0.5$ models a separable AR1×AR1 process. Cityblock metric may be appropriate when the dominant spatial processes are aligned with rows/columns as occurs in field experiments. Geometric anisotropy is discussed in most geostatistical books (Webster and Oliver, 2001, Diggle *et al.*, 2003) but rarely are the anisotropy angle or ratio estimated from the data. Similarly the smoothness parameter ν is often set a-priori (Kammann and Wand, 2003, Diggle *et al.*, 2003). However Stein (1999) and Haskard (2006) demonstrate that ν can be reliably estimated even for modest sized data-sets, subject to caveats regarding the sampling design.

The syntax for the Matérn class in ASReml is given by `MAT`k where k is the number of parameters to be specified; the remaining parameters take their default values. Use the `!G` qualifier to control whether a specified parameter is estimated or fixed. The order of the parameters in ASReml, with their defaults, is (ϕ, $\nu = 0.5$, $\delta = 1$, $\alpha = 0$, $\lambda = 2$). For

example, if we wish to fit a Matérn model with only ϕ estimated and the other parameters set at their defaults then we use `MAT1`. `MAT2` allows ν to be estimated or fixed at some other value (for example
`mat2(fac(xcoord,ycoord) !INIT 0.2 1.0 !GPF )`).

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 considering $0 \leq \alpha < \frac{\pi}{2}$ and $\delta > 0$, or by considering $0 \leq \alpha < \pi$ 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.

7.11.5 Notes on power models

Power models rely on the definition of distance for the associated term, for example,

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

Information for determining distances is supplied either implicitly by applying the model to the `fac()` of the coordinate variables, or explicitly with the `!COORD` qualifier.

- For one dimensional cases, either
  * `expv(fac(X))` where `X` contains the positions,
  * `expv(Trait !COORD x)` where x is a vector of positions.
- In two directions (`IEXP`, `IGAU`, `IEUC`, `AEXP`, `AGAU`, `MAT`n)
  * For a $\boldsymbol{G}$ structure relating to the model term `fac(`x`,`y`)`, use `fac(`x`,`y`)`. For example

```
⋮
yield ~ mu ...!r ieucv(fac(xcoord,ycoord !INIT 0.7 1.3)
```

7.11.6 Notes on Factor Analytic models

`FA`k, `FACV`k and `XFA`k are different parameterizations of the factor analytic model in which $\boldsymbol{\Sigma}$ is modelled as $\boldsymbol{\Sigma} = \boldsymbol{\Gamma}\boldsymbol{\Gamma}^{\mathsf{T}} + \boldsymbol{\Psi}$ where $\boldsymbol{\Gamma}^{(\omega \times k)}$ is a matrix of loadings on the covariance scale and $\boldsymbol{\Psi}$ is a diagonal vector of specific variances. See Smith *et al.* (2001) and Thompson *et al.* (2003) for examples of factor analytic models in multi-environment trials. The general limitations are

- that $\boldsymbol{\Psi}$ may not include zeros except in the `XFA`k formulation
- constraints are required in $\boldsymbol{\Gamma}$ for $k > 1$ for identifiability. These are automatically set unless the user formally constrains one parameter in the second column, two in the third

column, etc.

- the total number of estimated parameters $(k\omega+\omega-k(k-1)/2)$ may not exceed $\omega(\omega+1)/2$.

In `FA`k models the variance-covariance matrix $\boldsymbol{\Sigma}^{\,(\omega\times\omega)}$ is modelled on the correlation scale as $\boldsymbol{\Sigma} = \boldsymbol{DCD}$, where

- $\boldsymbol{D}^{\,(\omega\times\omega)}$ is diagonal such that $\boldsymbol{DD} = \text{diag}\,(\boldsymbol{\Sigma})\,$,

- $\boldsymbol{C}^{\,(\omega\times\omega)}$ is a correlation matrix of the form $\boldsymbol{FF}^{\mathsf{T}} + \boldsymbol{E}$ where $\boldsymbol{F}^{\,(\omega\times k)}$ is a matrix of loadings on the correlation scale and $\boldsymbol{E}$ is diagonal and is defined by difference,

- the parameters are specified in the order *loadings for each factor ($\boldsymbol{F}$) followed by the variances* $(\text{diag}\,(\boldsymbol{\Sigma}))$; when k is greater than 1, constraints on the elements of $\boldsymbol{F}$ are required, see Table 7.5,

`FACV`k models (`CV` for *covariance*) are an alternative formulation of `FA` models in which $\boldsymbol{\Sigma}$ is modelled as $\boldsymbol{\Sigma} = \boldsymbol{\Gamma\Gamma}^{\mathsf{T}} + \boldsymbol{\Psi}$ where $\boldsymbol{\Gamma}^{\,(\omega\times k)}$ is a matrix of loadings on the covariance scale and $\boldsymbol{\Psi}$ is diagonal. The parameters in `FACV`

- are specified in the order *loadings* ($\boldsymbol{\Gamma}$) *followed by variances* ($\boldsymbol{\Psi}$); when k is greater than 1, constraints on the elements of $\boldsymbol{\Gamma}$ are required, see Table 7.5,

- are related to those in `FA` by $\boldsymbol{\Gamma} = \boldsymbol{DF}$ and $\boldsymbol{\Psi} = \boldsymbol{DED}$,

`XFA`k (`X` for *extended*) is the third form of the factor analytic model and has the same parameterisation as for `FACV`, that is, $\boldsymbol{\Sigma} = \boldsymbol{\Gamma\Gamma}^{\mathsf{T}} + \boldsymbol{\Psi}$. However, `XFA` models

- have parameters specified in the order $\text{diag}\,(\boldsymbol{\Psi})$ and $\text{vec}(\boldsymbol{\Gamma})$; when k is greater than 1, constraints on the elements of $\boldsymbol{\Gamma}$ are required, see Table 7.5,

- may not be used in $\boldsymbol{R}$ structures,

- return the factors as well as the effects.

- permit some elements of $\boldsymbol{\Psi}$ to be fixed to zero,

- are computationally faster than the `FACV` formulation for large problems when k is much smaller than ω,

With multiple factors, some constraints are required to maintain identifiablity. Traditionally, this has simply been to set the leading loadings of new factors to zero. Loadings then need to be rotated to orthogonality. If no loadings are constrained, ASReml will rotate the loadings to orthogonality, after holding the loadings of lower factors fixed for a few iterations. The orthogonalization process occurs at the beginning of the iteration (so the final returned values have not been formally rotated).

Finding the REML solutions for multifactor Factor Analytic models can be difficult. The first problem is specifying initial values. When using `!CONTINUE` and progressing XFA(k) to XFA($k+1$), ASReml 3 initialises the factor $k+1$ at $\surd(\mathbf{\Psi} * 0.2)$, changing the sign of the (relatively) largest loading to negative. One strategy which sometimes works in this context is to hold the previously estimated factor loadings fixed for a few iterations so that the factor $k+1$ initally aims to explain variation previously incorporated in $\boldsymbol{\psi}$. Then allow all loadings to be updated in the remaining rounds. A second problem, at present unresolved but somewhat improved, is that sometimes the LogL rises to a relatively high value and then drifts away.

In an attempt to make the process easier, these two processes have been linked as an additional meaning for the `!AILOADING` n qualifier. When fitting k factors with $N > k$, the first $k-1$ loadings are held fixed (no rotation) for the first k iterations. Then for iterations $k+1$ to n, loadings vectors are updated in pairs, and rotated. If `!AILOADING` is not set by the user and the model is an upgrade from a lower order XFA, `!AILOADING` is set to 4.

The problem of XFA loadings going off-scale has been reduced by adding a variable penalty the the loading part of the AI matrix.

It is not unusual for users to have trouble comprehending and fitting extended factor analytic models, especially with more than two factors. Two examples are developed in a separate document available on request.

7.12 Variance models available in ASReml

Table 7.6: Details of the variance models available in ASReml

variance structure name variance model function name	description	algebraic form	number of parameters†		
			corr	hom variance	het variance
correlation models					
One-dimensional, equally spaced					
ID `id`	identity	$C_{ii} = 1,\ C_{ij} = 0,\ \ i \neq j$	0	1	ω
AR1 `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 `ar2`	2^{nd} order autoregressive	$C_{ii} = 1,$ $C_{i+1,i} = \phi_1/(1-\phi_2)$ $C_{ij} = \phi_1 C_{i-1,j} + \phi_2 C_{i-2,j},\ i > j+1$ $\lvert\phi_1\rvert < (1-\phi_2), \lvert\phi_2\rvert < 1$	2	3	$2 + \omega$
AR3 `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 `sar1`	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 `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	$2 + \omega$

7.12 Variance models available in ASReml

Details of the variance models available in ASReml

variance structure name	description	algebraic form	number of parameters†		
`variance model function name`			corr	hom variance	het variance
MA1 `ma1`	1^{st} order moving average	$C_{ii} = 1,$ $C_{i+1,i} = -\theta_1/(1+\theta_1^2)$ $C_{ji} = 0,\ j > i+2$ $\lvert\theta_1\rvert < 1$	1	2	$1+\omega$
MA2 `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 `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	$2+\omega$
CORU `coru`	uniform correlation	$C_{ii} = 1,\ C_{ij} = \phi,\ i \neq j$	1	2	$1+\omega$
CORB `corb`	banded correlation	$C_{ii} = 1$ $C_{i+j,i} = \phi_j,\ 1 \leq j \leq \omega-1$ $\lvert\phi_j\rvert < 1$	$\omega-1$	ω	$2\omega-1$
CORG `corg`	general correlation `CORGH` = `US`	$C_{ii} = 1$ $C_{ij} = \phi_{ij},\ i \neq j$ $\lvert\phi_{ij}\rvert < 1$	$\frac{\omega(\omega-1)}{2}$	$\frac{\omega(\omega-1)}{2}+1$	$\frac{\omega(\omega-1)}{2}+\omega$

7.12 Variance models available in ASReml

Details of the variance models available in ASReml

variance structure name	description	algebraic form	number of parameters†		
variance model function name			corr	hom variance	het variance
One-dimensional unequally spaced					
EXP exp	exponential	$C_{ii} = 1$ $C_{ij} = \phi^{\lvert x_i - x_j \rvert},\ i \neq j$ x_i are *coordinates* $0 < \phi < 1$	1	2	$1 + \omega$
GAU gau	gaussian	$C_{ii} = 1$ $C_{ij} = \phi^{(x_i - x_j)^2},\ i \neq j$ x_i are *coordinates* $0 < \phi < 1$	1	2	$1 + \omega$
Two-dimensional irregularly spaced					
		$\boldsymbol{x}$ and $\boldsymbol{y}$ vectors of coordinates $\theta_{ij} = \min(d_{ij}/\phi_1, 1)$ d_{ij} is euclidean distance			
IEXP iexp	isotropic exponential	$C_{ii} = 1$ $C_{ij} = \phi^{\lvert x_i - x_j \rvert + \lvert y_i - y_j \rvert},\ i \neq j$ $0 < \phi < 1$	1	2	$1 + \omega$
IGAU igau	isotropic gaussian	$C_{ii} = 1$ $C_{ij} = \phi^{(x_i - x_j)^2 + (y_i - y_j)^2},\ i \neq j$ $0 < \phi < 1$	1	2	$1 + \omega$
IEUC ieuc	isotropic euclidean	$C_{ii} = 1$ $C_{ij} = \phi^{\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}},\ i \neq j$ $0 < \phi < 1$	1	2	$1 + \omega$
LVR lvr	linear variance	$C_{ij} = (1 - \theta_{ij})$ $0 < \phi_1$	1	2	$1 + \omega$

7.12 Variance models available in ASReml

Details of the variance models available in ASReml

variance structure name	description	algebraic form	number of parameters†		
variance model function name			corr	hom variance	het variance
SPH sph	spherical	$C_{ij} = 1 - \frac{3}{2}\theta_{ij} + \frac{1}{2}\theta_{ij}^3$ $0 < \phi_1$	1	2	$1 + \omega$
CIR cir	circular (Webster & Oliver, 2001, p 113)	$C_{ij} = 1$ $- \frac{2}{\pi}(\theta_{ij}\sqrt{1-\theta_{ij}^2} + \sin^{-1}\theta_{ij})$ $0 < \phi_1$	1	2	$1 + \omega$
AEXP aexp	anisotropic exponential	$C_{ii} = 1$ $C_{ij} = \phi_1^{\lvert x_i - x_j \rvert}\phi_2^{\lvert y_i - y_j \rvert}$ $0 < \phi_1 < 1$, $0 < \phi_2 < 1$	2	3	$2+\omega$
AGAU agau	anisotropic gaussian	$C_{ii} = 1$ $C_{ij} = \phi_1^{(x_i - x_j)^2}\phi_2^{(y_i - y_j)^2}$ $0 < \phi_1 < 1$, $0 < \phi_2 < 1$	2	3	$2 + \omega$
MATk mat$k$	Matérn with first $1 \le k \le 5$ parameters spec-ified by the user	C_{ij} =Matérn: see text $\phi > 0$ range, ν shape(0.5) $\delta > 0$ anisotropy ratio(1), α anisotropy angle(0), $\lambda(1\|2)$ metric(2)	k	k+1	$k + \omega$
heterogeneous variance models					
DIAG diag	diagonal = IDH idh	$\mathbf{\Sigma}_{ii} = \phi_i$ $\mathbf{\Sigma}_{ij} = 0,\ i \neq j$	-	-	ω
US us	unstructured general covariance matrix	$\mathbf{\Sigma}_{ij} = \phi_{ij}$	-	-	$\frac{\omega(\omega+1)}{2}$
OWNk own$k$	user explicitly forms $\boldsymbol{V}$ and $\partial\boldsymbol{V}$		-	-	k

7.12 Variance models available in ASReml

Details of the variance models available in ASReml

variance structure name	description	algebraic form	number of parameters[†]		
`variance model function name`			corr	hom variance	het variance
ANTE1 `ante1` ANTEk `ante`k	1^{st} k order k^{th} antedependence $1 \leq k \leq \omega - 1$	$\boldsymbol{\Sigma}^{-1} = \boldsymbol{UDU}^{\mathsf{T}}$ $D_{ii} = d_i,\ D_{ij} = 0,\ i \neq j$ $U_{ii} = 1,\ U_{ij} = u_{ij},\ 1 \leq j - i \leq k$ $\boldsymbol{U}_{ij} = 0,\ i > j$	-	-	$\frac{\omega(\omega+1)}{2}$
CHOL1 `chol1` CHOLk `chol`k	1^{st} k order k^{th} cholesky $1 \leq k \leq \omega - 1$	$\boldsymbol{\Sigma} = \boldsymbol{LDL}^{\mathsf{T}}$ $D_{ii} = d_i,\ D_{ij} = 0,\ i \neq j$ $L_{ii} = 1,\ L_{ij} = l_{ij},\ 1 \leq i - j \leq k$	-	-	$\frac{\omega(\omega+1)}{2}$
FA1 `fa1` FAk `fa`k	1^{st} k order k^{th} factor analytic	$\boldsymbol{\Sigma} = \boldsymbol{DCD}$, $\boldsymbol{C} = \boldsymbol{FF}^{\mathsf{T}} + \boldsymbol{E}$, $\boldsymbol{F}$ contains k correlation factors $\boldsymbol{E}$ diagonal $\boldsymbol{DD} = \text{diag}\,(\boldsymbol{\Sigma})$	-	-	$\omega + \omega$ $k\omega + \omega$
FACV[1] `facv1` FACVk `facv`k	1^{st} k order k^{th} factor analytic covariance form	$\boldsymbol{\Sigma} = \boldsymbol{\Gamma\Gamma}^{\mathsf{T}} + \boldsymbol{\Psi}$, $\boldsymbol{\Gamma}$ contains covariance factors $\boldsymbol{\Psi}$ contains specific variance	-	-	$\omega + \omega$ $k\omega + \omega$
XFA1 `xfa1` XFAk `xfa`k	1^{st} k order k^{th} extended factor analytic	$\boldsymbol{\Sigma} = \boldsymbol{\Gamma\Gamma}^{\mathsf{T}} + \boldsymbol{\Psi}$, $\boldsymbol{\Gamma}$ contains covariance factors $\boldsymbol{\Psi}$ contains specific variance	-	-	$\omega + \omega$ $k\omega + \omega$

7.12 Variance models available in ASReml

Details of the variance models available in ASReml

variance structure name	description	algebraic form	number of parameters[†]		

`variance`
`model`
`function`
`name`
			corr	hom
				variance
					het
					variance

relationship matrices[‡]					
`AINV`	inverse relationship matrix derived from pedigree		0	1	-
NRM `nrm`	relationship matrix derived from pedigree		0	1	-
GIV1 `giv1`	generalized inverse number 1		0	1	-
⋮	⋮	⋮	⋮	⋮	
GIV8 `giv8`	generalized inverse matrix 8		0	1	-
GRM1 `grm1`	generalized relationship number 1		0	1	-
⋮	⋮	⋮	⋮	⋮	
GRM8 `grm8`	generalized relationship matrix 8		0	1	-

[†] This is the number of variance structure parameters, ω is the dimension of the matrix. The homogeneous variance form is specified by appending `V` to the correlation basename; the heterogeneous variance form is specified by appending `H` to the correlation basename

[‡] These will be associated with 1 variance parameter unless used in direct product with another structure that provides the variance. Appending a `v` to a name makes it explicit that a variance parameter is fitted.

8 Command file: Multivariate analysis

8.1 Introduction

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

8.1.1 Repeated measures on rats

Wolfinger (1996) summarises a range of variance structures that can be fitted to repeated measures data and demonstrates the models using five weights taken weekly on 27 rats subjected to 3 treatments. This command file demonstrates a multivariate analysis of the five repeated measures. Note that the two dimensional structure for residual errors meets the requirement of independent units and corresponds to the data being ordered traits within units.

```
Wolfinger rat data
 treat !A
 wt0 wt1 wt2 wt3 wt4
rat.dat
wt0 wt1 wt2 wt3 wt4 ~ Trait,
treat Trait.treat
residual id(units).us(Trait !GP)
```

8.1.2 Wether trial data

Three key traits for the Australian wool industry are the weight of wool grown per year, the cleanness and the diameter of that wool. Much of the wool is produced from wethers and most major producers have traditionally used a particular strain or *bloodline*. To assess the importance of bloodline differences, many wether trials were conducted. One trial, conducted from 1984 to 1988 at Borenore near Orange, involved 35 teams of wethers representing 27 bloodlines. The file `wether.dat` shown below contains greasy fleece weight (kg), yield (percentage of clean fleece weight to greasy fleece weight) and fibre diameter (microns). The code (`wether.as`) to the right performs a basic bivariate analysis of this data.

```
Orange Wether Trial 1984-8
 SheepID !I
 TRIAL
 BloodLine !I
 TEAM * YEAR *
 GFW YLD FDIAM
wether.dat !skip 1
GFW FDIAM ~ Trait Trait.YEAR,
!r us(Trait).id(TEAM) us(Trait).id(SheepID)
residual id(units).us(Trait !GP)
predict YEAR Trait
```

```
SheepID Site Bloodline Team Year GFW Yield FD
0101 3 21 1 1 5.6 74.3 18.5
0101 3 21 1 2 6.0 71.2 19.6
0101 3 21 1 3 8.0 75.7 21.5
0102 3 21 1 1 5.3 70.9 20.8
0102 3 21 1 2 5.7 66.1 20.9
0102 3 21 1 3 6.8 70.3 22.1
0103 3 21 1 1 5.0 80.7 18.9
0103 3 21 1 2 5.5 75.5 19.9
0103 3 21 1 3 7.0 76.6 21.9
⋮
4013 3 43 35 1 7.9 75.9 22.6
4013 3 43 35 2 7.8 70.3 23.9
4013 3 43 35 3 9.0 76.2 25.4
4014 3 43 35 1 8.3 66.5 22.2
4014 3 43 35 2 7.8 63.9 23.3
4014 3 43 35 3 9.9 69.8 25.5
4015 3 43 35 1 6.9 75.1 20.0
4015 3 43 35 2 7.6 71.2 20.3
4015 3 43 35 3 8.5 78.1 21.7
```

8.2 Model specification

The syntax for specifying a multivariate linear model in ASReml is

Y-variates ~ *fixed* [`!r` *conrandom*] [`!f` *sparse_fixed*]
[`residual` *conresidual*]

- *Y-variates* is a list of up to 20 traits (there may be more than 20 actual variates if the list includes sets of variates defined with `!G` on page 49),

- *fixed*, *conrandom* and *sparse_fixed* are as in the univariate case (see Chapter 6) but involve the special term `Trait` and interactions with `Trait`.

  The design matrix for `Trait` has a level (column) for each trait.

  - `Trait` by itself fits the mean for each variate,
  - in an interaction `Trait.Fac` fits the factor `Fac` for each variate and `Trait.Cov` fits the covariate `Cov` for each variate. An explanatory factor or covariable associated with Trait `i` can be fitted using `at(Trait,i).Fac` or `at(Trait,i).Cov`.

ASReml internally arranges the data so that n data records containing t traits each becomes n sets of t analysis records indexed by the internal factor `Trait` ie. nt analysis records ordered `Trait` within data record. If the data is already in this long form, use the `!ASMV` t qualifier to indicate that a multivariate analysis is required.

8.3 Residual variance structures

Using the notation of Section 2.1.11, consider a multivariate analysis with t traits and n units in which the data are ordered *traits* within *units*. An algebraic expression for the residual variance matrix in this case is

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

where $\boldsymbol{\Sigma}^{(t \times t)}$ is an unstructured variance matrix. This is the general form of residual variance structures required for multivariate analysis.

8.3.1 Specifying multivariate variance structures in ASReml

A standard multivariate analysis is achieved using the the `us()` variance model function for the two random `Trait` components, and specifying the R structure for the residual error term as `residual id(units).us(Trait)`.

```
Orange Wether Trial 1984-8
 SheepID !I
 TRIAL
 BloodLine !I
 TEAM *
 YEAR *
 GFW YLD FDIAM
wether.dat !skip 1
GFW FDIAM ~ Trait Trait.YEAR,
!r us(Trait).id(TEAM) us(Trait).id(SheepID)
residual id(units).us(Trait !GP)
```

- if provided, the initial values are for the lower triangle of the (symmetric) matrix specified row-wise,
- finding reasonable initial values can be a problem. When no initial values are provided (as in code box), ASReml takes half of the phenotypic variance matrix of the data as an initial value.

Since the variance component matrices for the `TEAM` and `SheepID` strata are not specified, ASReml will plug in values derived from the observed phenotypic variance matrix. `!GP` requests that the resulting estimated matrix be kept within the parameter space, ie. it is to be positive definite:

8.3 Residual variance structures

The special qualifiers relating to multivariate analysis are `!ASUV` and `!ASMV` *t*, see Table 5.4 for details:

- to use an error structure other than US for the residual stratum you must also specify `!ASUV` (see Table 5.4) and include `mv` in the model if there are missing values,

- to perform a multivariate analysis when the data have already been expanded use `!ASMV` *t* (see Table 5.4)

- *t* is the number of traits that `ASReml` should expect

- the data file must have *t* records for each multivariate record although some may be coded missing.

Note that, if no `residual` line is inserted the `id(units).us(Trait)` variance structure is assumed for multivariate data.

9 Command file: Genetic analysis

9.1 Introduction

In genetic analysis using an 'animal model' or 'sire model', we have data on subjects that are genetically related. The relationships are defined via a pedigree. The subject effects are therefore correlated and, assuming normal modes of inheritance, the correlation expected from additive effects can be computed from the pedigree provided all the direct links are in the pedigree. The matrix of such relationships is called the numerator relationship matrix. It is actually the *inverse* relationship matrix that is required for analysis and that is formed by ASReml. Users new to this subject might find notes Mixed Models for Genetic analysis [1] by Julius van der Werf helpful.

For the more general situation where the pedigree based relationship matrix is not the appropriate/required matrix, the user can provide a general relationship matrix (GRM) matrix explicitly in a .grm file, or its inverse in a .giv file.

As an example for this chapter, we consider data presented in Harvey (1977) using the command file harvey.as.

9.2 The command file

In ASReml the !P data field qualifier indicates that the corresponding data field has an associated pedigree. The file containing the pedigree (harvey.ped in the example) for animal is specified after all field definitions and before the datafile definition. See below for the first 20 lines of harvey.ped together with the corresponding lines of the data file harvey.dat. All individuals appearing in the data file must appear in the pedigree file. When all the pedigree information (individual, male_parent, female_parent) appears as the first three fields

```
Pedigree file example
 Animal !P
 Sire !A
 Dam
 Line 2
 AgeOfDam
 adailygain
 Y2
 Y3
harvey.ped !ALPHA
harvey.dat
adailygain ∼ mu Line,#fixed model
!r nrmv(Animal !INIT 0.25)#random model
residual idv(units)
```

[1] http://www.vsni.co.uk/products/asreml/user/geneticanalysis.pdf

of the data file, the data file can double as the pedigree file. In this example the line `harvey.ped !ALPHA` could be replaced with `harvey.dat !ALPHA`. Often the pedigree file will include individuals for which there is no data, individuals that define genetic links between individuals with data. The `nrm` in `nrmv(Animal)` indicates that an additive (or numerator) relationship matrix (nrm) variance structure is constructed from the pedigree associated with `Animal`. The `v` in `nrmv` indicates that the nrm matrix is scaled by a variance parameter.

9.3 The pedigree file

The pedigree file is used to construct the genetic relationships for fitting a genetic animal model and is required if the `!P` qualifier is associated with a data field. The pedigree file:

- has three fields; the identities of an individual and its parents (or sire and maternal grand sire if the `!MGS` qualifier is specified (Table 9.1), Typically for animals, the male parent is listed first, but for trees, the mother tree may be first.

- an optional fourth field may supply inbreeding/selfing information used if the `!FGEN` qualifier is specified (Table 9.1),

- an additional field specifying the sex of the individual is required if the `!XLINK` qualifier is specified (Table 9.1),

- is ordered by generation so that the line giving the pedigree of an individual appears above any line where that individual appears as a parent,

- is read free format; it may be the same file as the data file if the data file is free format and has the necessary identities in the first three fields, see below,

- is specified on the line immediately after all field definitions and before the data file line in the command file,

- use `0` or `*` to represent unknown parents.

harvey.ped

```
101 SIRE_1 0
102 SIRE_1 0
103 SIRE_1 0
104 SIRE_1 0
105 SIRE_1 0
106 SIRE_1 0
107 SIRE_1 0
108 SIRE_1 0
109 SIRE_2 0
110 SIRE_2 0
111 SIRE_2 0
112 SIRE_2 0
113 SIRE_2 0
114 SIRE_2 0
115 SIRE_2 0
116 SIRE_2 0
117 SIRE_3 0
118 SIRE_3 0
119 SIRE_3 0
120 SIRE_3 0
⋮
```

harvey.dat

```
101 SIRE_1 0 1 3 192 390 2241
102 SIRE_1 0 1 3 154 403 2651
103 SIRE_1 0 1 4 185 432 2411
104 SIRE_1 0 1 4 183 457 2251
105 SIRE_1 0 1 5 186 483 2581
106 SIRE_1 0 1 5 177 469 2671
107 SIRE_1 0 1 5 177 428 2711
108 SIRE_1 0 1 5 163 439 2471
109 SIRE_2 0 1 4 188 439 2292
110 SIRE_2 0 1 4 178 407 2262
111 SIRE_2 0 1 5 198 498 1972
112 SIRE_2 0 1 5 193 459 2142
113 SIRE_2 0 1 5 186 459 2442
114 SIRE_2 0 1 5 175 375 2522
115 SIRE_2 0 1 5 171 382 1722
116 SIRE_2 0 1 5 168 417 2752
117 SIRE_3 0 1 3 154 389 2383
118 SIRE_3 0 1 4 184 414 2463
119 SIRE_3 0 1 5 174 483 2293
120 SIRE_3 0 1 5 170 430 2303
⋮
```

9.4 Reading in the pedigree file

The syntax for specifying a pedigree file in the ASReml command file is

pedigree_file [*qualifiers*]

- the *qualifiers*[2] are listed in Table 9.1,
- the identities (*individual*, *parent_1*, *parent_2*) are merged into a single list and the inverse relationship is formed before the data file is read,
- *parent_1* is typically male for animal pedigrees (sire) but often female for plant pedigrees; it must be the XY parent if the `!XLINK` qualifier is specified,
- when the data file is read, data fields with the `!P` qualifier are recoded according to the combined identity list,
- the inverse relationship matrix is automatically associated with factors coded from the pedigree file unless some other covariance structure is specified. The inverse relationship matrix is specified with the variance model name `NRM`, the variance model function name `nrm()`,
- the inverse relationship matrix is written to `ainverse.bin`,

[2] `http://www.vsni.co.uk/products/asreml/user/PedigeeNotes.pdf` contains details of these options.

  - if `ainverse.bin` already exists ASReml assumes it was formed in a previous run and has the correct inverse
    - `ainverse.bin` is read, rather than the inverse being reformed (unless `!MAKE` is specified); this saves time when performing repeated analyses based on a particular pedigree,
    - delete `ainverse.bin` or specify `!MAKE` if the pedigree is changed between runs,

- identities are printed in the `.sln` and the `.aif` file,
  - identities should be whole numbers less than 200,000,000 unless `!ALPHA` is specified,
  - pedigree lines for parents must precede their progeny,
  - unknown parents should be given the identity number 0,
  - if an individual appearing as a parent does not appear in the first column, it is assumed to have unknown parents, that is, parents with unknown parentage do not need their own line in the file,
  - identities may appear as both male and female parents, for example, in forestry.

We refer the reader to the sheep genetics example on page 318.

Table 9.1: List of pedigree file qualifiers

qualifier	description
`!ALPHA`	indicates that the identities are alphanumeric with up to 225 characters; otherwise by default they are numeric whole numbers < 200,000,000. If using long alphabetic identities, use `!SLNFORM` to see the full identity in the `.sln` file.
`!DIAG` `!AIF`	causes the pedigree identifiers, the diagonal elements of the Inverse of the Relationship and the inbreeding coefficients for the individuals (calculated as the diagonal of $\boldsymbol{A}-\boldsymbol{I}$), and a factor with levels `Parent` and `Nonparent` indicating if the individual is a parent (with progeny in the pedigree) or a non-parent (with no progeny in the pedigree) to be written to *basename*`.aif`.
`!FGEN` [f]	indicates the pedigree file has a fourth field containing the level of selfing or the level of inbreeding in a base individual. In the fourth field, 0 indicates a simple cross, 1 indicates selfed once, 2 indicates selfed twice, etc.. A value between 0 and 1 for a base individual is taken as its inbreeding value. If the pedigree has implicit individuals (they appear as parents but not in the first field of the pedigree file), they will be assumed base non-inbred individuals unless their inbreeding level is set with `!FGEN` f where $0 < f < 1$ is the inbreeding level of such individuals. Individuals with one or both parents unknown, and without a specific non-zero inbreeding coefficient provided in the fourth filed of the pedigree, will are assigned an inbreeding coefficient f.

9.4 Reading in the pedigree file

List of pedigree file qualifiers

qualifier	description
!GIV !GIV 2	instructs ASReml to write out the A-inverse in the format of .giv files. !GIV 2 writes the pedigree of the parents to *basename*_Parent.ped and the diagonal elements of the A-inverse to *basename*_Q.giv with offspring identifiers (see Section 9.7). If !GROUPS is also specified, this .giv file will include the !GROUPSDF qualifier on its first line.
!GOFFSET *o*	An alternative to group constraints (see !GROUPS below) is to shrink the group effects by adding the constant o (> 0) to the diagonal elements of $\boldsymbol{A}^{-1}$ pertaining to groups. When a constant is added, no adjustment of the degrees of freedom is made for genetic groups. Use !GOFFSET -1 to add no offset but to suppress insertion of constraints where empty groups appear. The empty groups are then not counted in the DF adjustment.
!GROUPS *g*	includes genetic groups in the pedigree. The first g lines of the pedigree identify genetic groups (with zero in both parent fields). All other lines must specify one of the genetic groups as parent if the actual parent is unknown. You may insert groups identifiers with no members to define constraints on groups, that is to associate groups into supergroups where the supergroup fixed effect is formally fitted separately in the model. A constraint is added to the inverse which causes the preceding set of groups which have members to have effects which sum to zero. The issue is to get the degrees of freedom correct and to get the correct calculation of the Likelihood, especially in bivariate cases where DF associated with groups may differ between traits. The !LAST qualifier (see page 80) is designed to help as without it, reordering may associate singularities in the $\boldsymbol{A}$ matrix with random effects which at the very least is confusing. When the $\boldsymbol{A}$ matrix incorporates fixed effects, the number of DF involved may not be obvious, especially if there is also a sparsely fitted fixed HYS factor. The number of Fixed effects (degrees of freedom) associated with GROUPS is taken as the declared number less twice the number of constraints applied. This assumes all groups are represented in the data, and that degrees of freedom associated with group constraints will be fitted elsewhere in the model.
!INBRED	Each cross is assumed to be selfed several times to stabilize as an inbred line as is usual for cereals such as wheat, before being evaluated or crossed with another line. Since inbreeding is usually associated with strong selection, it is not obvious that a pedigree assumption of covariance of 0.5 between parent and offspring actually holds. Do not use the !INBRED qualifier with the !MGS or !SELF qualifiers.
!LONGINTEGER	indicates the identifiers are numeric integer with less than 16 digits. The default is integer values with less than 9 digits. The alternative is alphanumeric identifiers with up to 255 character indicated by !ALPHA.
!MAKE	forces ASReml to make the A-inverse (rather than trying to retrieve it from the ainverse.bin file).
!MEUWISSEN	The default method for forming $\boldsymbol{A}^{-1}$ is based on the algorithm of Meuwissen and Luo (1992).
!MGS	indicates that the third identity is the sire of the dam rather than the dam.
!QUAAS	The original routine for calculating $\boldsymbol{A}^{-1}$ in ASReml was based on Quaas (1976)
!REPEAT	tells ASReml to ignore repeat occurrences of lines in the pedigree file. **Warning** Use of this option will avoid the check that animals occur in generational order, but generational order is still required.

List of pedigree file qualifiers

qualifier	description
`!SARGOLZAEI`	an alternative procedure for computing $\boldsymbol{A}^{-1}$ was developed by Sargolzaei *et al.*(2005).
`!SELF` *s*	allows partial selfing when second parent is unknown. It indicates that progeny from a cross where the second parent (male_parent) is unknown, is assumed to be from selfing with probability s and from outcrossing with probability $(1 - s)$. This is appropriate in some forestry tree breeding studies where seed collected from a tree may have been pollinated by the mother tree or pollinated by some other tree (Dutkowski and Gilmour, 2001). Do not use the `!SELF` qualifier with the `!INBRED` or `!MGS` qualifiers.
`!SKIP` *n*	allows you to skip n header lines at the top of the file.
`!SORT`	causes ASReml to sort the pedigree into an acceptable order, that is parents before offspring, before forming the A-Inverse. The sorted pedigree is written to a file whose name has `.srt` appended to its name.
`!XLINKR`	requests the formation of the (inverse) relationship matrix for the X chromosome as described by Fernando and Grossman (1990) where the first parent is XY and the second is XX. This NRM inverse matrix is formed in addition to the usual $\boldsymbol{A}^{-1}$ and can be accessed as `GRM1` or as specified in the output. The pedigree must include a fourth field which codes the SEX of the individual. The actual code used is up to the user and deduced from the first line which is assumed to be a an XY individual. Thus, whatever string is found in the fourth field on the first line of the pedigree is taken to mean XY and any other code found on other records is taken to mean XX.

9.5 Genetic groups

If all individuals belong to one genetic group, then use `0` as the identity of the parents of base individuals. However, if base individuals belong to various genetic groups this is indicated by the `!GROUPS` qualifier and the pedigree file must begin by identifying these groups. All base individuals should have group identifiers as parents. In this case the identity `0` will only appear on the group identity lines, as in the following example where three sire lines are fitted as genetic groups.

```
Genetic group example
 Animal !P
 Sire !A
 Dam
 Line 2
 AgeOfDam
 adailygain
 Y2
 Y3
harveyg.ped !ALPHA !GROUPS 3
harvey.dat
adailygain ~ mu Line, # fixed model
!r grm1v(Animal !INIT 0.25)) # random model
residual idv(units)
```

```
G1 0 0
G2 0 0
G3 0 0
SIRE_1 G1 G1
SIRE_2 G1 G1
SIRE_3 G1 G1
SIRE_4 G2 G2
SIRE_5 G2 G2
SIRE_6 G3 G3
SIRE_7 G3 G3
SIRE_8 G3 G3
SIRE_9 G3 G3
101 SIRE_1 G1
102 SIRE_1 G1
103 SIRE_1 G1
⋮
163 SIRE_9 G3
164 SIRE_9 G3
165 SIRE_9 G3
```

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

9.6 Reading a user defined (inverse) relationship matrix

Sometimes a relationship matrix is required other than the one ASReml can produce from the pedigree file. We call this a GRM (General Relationship Matrix). The inverse of a GRM is a GIV matrix. The user can provide the relationship matrix in a `.grm` file and ASReml will invert it to form the GIV matrix (since it is the inverse that is used in the mixed model equations). Alternatively, the user can provide a `.giv` file containing the inverted GRM matrix.

The syntax for specifying a GRM file (say *name*`.grm`) or the GIV file (say *name*`.giv`) is

name`.[s|d]grm` [`!SKIP` n] [`!DENSEGRM` [o]] [`!GROUPDF` n] [`!ND|!PSD|!NSD`] [`!PRECISION` n]
or
name`.[s|d]giv` [`!SKIP` n] [`!DENSEGIV` [o]] [`!GROUPDF` n] [`!SAVEGIV` f]

- the named file must have a `.giv`, `.grm`, `.sgiv`, `.sgrm`, `.dgiv` or `.dgrm` extension,
- `.sgiv` and `.sgrm` files are binary format files and will be read lower triangle row-wise assuming single precision,
- `.dgiv` and `.dgrm` files are binary format files and will be read lower triangle row-wise assuming double precision,
- the named file will be read assuming single/double precision lower triangle row-wise,

- the G (inverse) files must be specified on the line(s) immediately prior to the data file line after any pedigree file,
- up to 98 G (inverse) matrices may be defined,
- the file must be in SPARSE format unless the `!DENSE` qualifier is specified,
- a dense format file has the matrix presented lower triangle rowwise, with each row beginning on a new line,
- a sparse format file must be free format with three numbers per line, namely

  row column value

  defining the lower triangle row-wise of the matrix,
- the file must be sorted *column* within *row*,
- every diagonal element must be present; missing off-diagonal elements are assumed to be zero cells,
- the file is used by associating it with a factor in the model. The number and order of the rows must agree with the size and order of the associated factor,

```
1 1 1
2 2 1
3 3 1
4 4 1
5 5 1.0666667
6 5 -0.2666667
6 6 1.0666667
7 7 1.0666667
8 7 -0.2666667
8 8 1.0666667
9 9 1.0666667
10 9 -0.2666667
10 10 1.0666667
11 11 1.0666667
12 11 -0.2666667
12 12 1.0666667
```

- the `!SKIP` n qualifier tells ASReml to skip n header lines in the file.

The `.giv` file presented in the code box gives the G inverse matrix on the right

$$\begin{bmatrix} \boldsymbol{I}_4 & \mathbf{0} \\ \mathbf{0} & \boldsymbol{I}_4 \otimes \begin{bmatrix} 1.067 & -0.267 \\ -0.267 & 1.067 \end{bmatrix} \end{bmatrix}$$

The easiest way to ensure the variable is coded to match the order of the GRM file is to supply a list of level names in the variable definition. For example, `genotype !A !L Gorder.txt` would code the variable `genotype` to agree with the order of level names present in the file `Gorder.txt` which would be the order used in creating the GRM/GIV matrix.

If the file has a `.grm` file extension, ASReml will invert the GRM matrix. If it is not Positive Definite, the job will abort unless an appropriate qualifier `!ND`, `!PSD` or `!NSD` is supplied. These qualifiers do not modify the matrix, they just instruct ASReml to proceed regardless. If the matrix has positive and negative eigenvalues, `!ND` instructs ASReml to ignore the condition and proceed anyway. If the matrix is positive semi-definite (positive and zero eigenvalues), `!PSD` allows ASReml to introduce Lagrangian multipliers to accommodate linear dependencies and rows with zero elements, and allows `ASReml` to proceed. Linear dependencies occur, for example, when the list of individuals includes clones. Rows with zero elements occur when the GRM represents a dominance matrix, and the list of individuals includes fully inbred individuals which, by definition, have zero dominance variance. If the matrix has positive, zero and negative eigenvalues, !NSD may be used to allow ASReml to continue. The zero eigenvalues are handled as for !PSD. Sometimes, with negative eigenvalues, the iteration sequence may fail as some parameter values will result in a negative residual sum of squares.

If the specified .giv file does not exist but there is a .grm file of the same name, ASReml will read and invert the .grm file, and write the inverse to the .giv file if !SAVEGIV [*f*] is specified. Its is written in DENSE format unless $f = 1$. !SAVEGIV 3 writes the GIV matrix as an .sgiv file. !SAVEGIV 4 writes the GIV matrix as a .dgiv file, where .sgiv is a single precision lower triangle row-wise binary file and .dgiv is a double precision lower triangle row-wise binary file. !PRECISION *n* changes the value used to declare a singularity when inverting a GRM file from 1D-7 to 1D-n.

A GRM can be associated with a factor i by using the variance model function grm*i*(*f*); which associates the *i*th GRM with factor *f*, for example,

```
grm1v(animal !INIT 0.12)
```

or

```
coruh(site).grm2(variety)
```

It is imperative that the GIV/GRM matrix be defined with the correct row/column order, the order that matches the order of the levels in the factor it is associated with. The easiest way to check this is to compare the order used in the GIV/GRM file with the order reported in the .sln file when the model is fitted.

Another example of !L (Section 5.4.1) is in analysis on data with 2 relationship matrices based on two separate pedigrees. ASReml only allows one pedigree file to be specified but can create an inverse relationship matrix and store the result in a GIV file. So, 2 relationship matrices based on two separate pedigrees may be used by generating a GIV file from one pedigree and then using that GIV file and the other pedigree in a subsequent run. To process the GIV file properly, we must also generate a file with identities as required for the GIV matrix. An example of this is if the file Hybrid.as includes

```
!PART 1
Mline !P
Fline !A
...
Mline.ped !GIV !DIAG #!GIV generates the file Hybrid1A.giv and !DIAG
#generates Hybrid1.aif which contains the identifier names
!PART 2 #reads in inverse additive relationship matrix generated in !PART 1
Mline !A !L Hybrid1.aif !SKIP 1#associates identifier names with levels of Mline
#used in giv file
Fline !P
...
```

```
Fline.ped !GIV !DIAG
Hybrid1_A.giv #formed in part 1 from Mline.ped
Hybrid.asd !SKIP 1
...
...  grm1(Mline) nrm(Fline) #using new synonyms and functions
```

9.6.1 Genetic groups in GIV matrices

If a user creates a GIV file outside ASReml which has fixed degrees of freedom associated with it, a `!GROUPSDF` n qualifier is provided to specify the number of fixed degrees of freedom (n) incorporated into the GIV matrix. The `!GROUPSDF` qualifier is written into the first line of the `.giv` matrix produced by the `!GIV` qualifier of the pedigree line if the pedigree includes genetic groups, and will be honoured from there, when reusing a GIV matrix formed from a pedigree with genetic groups in ASReml.

When groups are constrained, then it will be the number of groups less number of constraints. For example, if the pedigree file qualified by `!GROUPS 7` begins

```
A 0 0
B 0 0
C 0 0
ABC 0 0 # ABC is not present in the subsequent pedigree lines
D 0 0
E 0 0
DE 0 0 # DE is not present in the subsequent pedigree lines
```

there are actually only 5 genetic groups and two constraints so that the fixed effects for `A`, `B` and `C` sum to zero, and for `D` and `E` sum to zero leaving only 3 fixed degrees of freedom fitted. Therefore if the $\boldsymbol{A}$ inverse for this pedigree was saved, it will contain `!GROUPSDF 3` in the GIV file.

9.6.2 The example continued

Below is an extension of `harvey.as` to use `harvey.giv` which is partly shown to the right. This G inverse matrix is an identity matrix of order 74 scaled by `0.5`, that is, $0.5\boldsymbol{I}_{74}$. This model is simply an example which is easy to verify. Note that `harvey.giv` is specified on the line immediately preceding `harvey.dat`.

command file   .giv file

```
giv file example
 Animal !P
 Sire !P
 Dam
 Line 2
 AgeOfDam
 adailygain
 Y2
 Y3
harvey.ped !ALPHA
harvey.giv # giv structure file
harvey.dat
adailygain ~ mu Line, # fixed model
!r grm1v(Sire !INIT 0.25) # random model
residual idv(units)
```

```
01 01 .5
02 02 .5
03 03 .5
04 04 .5
05 05 .5
.
.
.
72 72 .5
73 73 .5
74 74 .5
```

Model term specification associating the `harvey.giv` structure to the coding of sire takes precedence over the relationship matrix structure implied by the `!P` qualifier for sire. In this case, the `!P` is being used to amalgamate animals and sires into a single list, and the `.giv` matrix must agree with the list order.

9.7 The reduced animal model (RAM)

The reduced animal model was devised to reduce the computation involved in fitting a large animal model. When there is at most one record per individual, a large proportion of the individuals are non-parents and have no progeny and there is interest in predictions for parents alone. This can happen in large forestry trials. The reduced animal model expresses the non-parent genetic effect in terms of parent effects and a Mendelian sampling term that is combined with the residual effect for the residual. We consider the case when there is data on parents and non-parents and some individuals are inbred.

An example tree model for a single trait and a single site might be

```
DBH ~ mu !r nrmv(tree) plot ar1v(column).ar1(row)
residual idv(units)
```

since trees are often planted in plots of say 5 trees. This is a spatial analysis; the `idv(units)` term is required so that error variance is not transferred to the `nrmv(tree)` term since trees are unreplicated.

This analysis requires a pedigree file, say `TreePed.csv`, and if the `!DIAG` qualifier is specified on the pedigree line, the resulting `.aif` file will contain the inbreeding level for every tree in the pedigree, the diagonal of the $\boldsymbol{A}^{-1}$ matrix and a `N/P` code distinguishing parents (with progeny) from non-parents (without progeny).

To analyse the data using the RAM, we need to incorporate these last two columns into the data file (which can be done with the `!MERGE` statement). If there is data on parents, further processing of the data file is required: create a copy of the 'tree' field, call it say 'parent', and change it to '0' for the progeny records.

Assume our data file `ramdbh.txt` has fields `tree mum dad row column plot DBH AIdiag OP parent` and we have deleted the non-parent rows from the full pedigree file to form `ParentPed.txt`. If you have a pedigree file for all trees, processing that pedigree with the `!GIV 2` qualifier will create a pedigree file just containing the parents and also the `Q.giv` file for the non-parent referred to below. If we assume a heritability of 0.1111 so that the ratio of genetic variance to residual variance is 0.125, the following model will estimate the breeding values for the parents directly:

```
RAM BLUP model
 tree !
 mum !P !*V21
 dad !P !*V21
 row *
 column *
 plot *
 DBH
 AIdiag !*V21
 NP !A !L Nonparent Parent
 parent !P
 filter !=NP !==1 # create Nonparent filter
 mum !*filter
 dad !*filter
 AIdiag !*filter
 WT !=0.125 !+AIdiag !^-1 !*AIdiag !+1 !-filter
ParentPed.txt
ramdbh.txt
DBH !WT WT ~ mu,
!r str(parent and(mum,0.5) and(dad.0.5) id(1).nrmv(parent !0.125)),
plot ar1v(column).ar1(row)
residual idv(units)
```

In this model,

- `NP !A !L Nonparent Parent` ensures the `NP` data field is coded 1 for non-parents and 2 for parents.
- `filter !=NP !==1` creates a variable that is 1 for non-parents and zero for parents.
- The `!*filter` transformations put `mum`, `dad` and `AIdiag` information to zero for parents.
- `WT !=0.125 !+AIdiag !^-1 !*AIdiag !+1 !-V21` creates a weight variable which is 1 for parent records, $q/(q+\gamma)$ for a non-parent record with q the respective diagonal element

of `AIdiag`, with $q = 2$ for non-inbred non-parents, and γ is the variance ratio σ_g^2/σ_e^2, 0.125 in this case. This weighting corresponds to a residual variance for a non-parent record of $(\sigma_g^2/q) + \sigma_e^2$.

- If there is no direct information on parents, the parent term is replaced by zero, where zero is a variable with zero elements.

- If `dad` is unknown, the `and(dad)` term is dropped.

- The BLUPs of a non-parent will need to be calculated outside ASReml by adding $[\gamma/(q+\gamma)]$ times its residual to the average of the parental BLUPs.

Prediction of parental values with assumed heritability was the main motivation for the development of the reduced animal model. Estimation of genetic variance parameters is a little more complicated and the computational gains of removing non-parent genetic values from the estimation procedure only apply if it is reasonable to form a small number of groups with roughly similar `AIdiag` values. If `AIG` is this group factor then one can estimate residual variances in each group using `sat(AIG).idv(units)` and use the variance parameter linear model facilities to constrain the residual variances and the parent variance to be a function of the genetic and residual variances.

9.8 Factor effects with large Random Regression models

One use of the GRM matrix is to allow more computationally efficient fitting of random regression models associating $\boldsymbol{u}$, a vector of f factor effects with $\boldsymbol{v}$ a vector of m regression effects through the model $\boldsymbol{u} = \boldsymbol{Mv}$ where the matrix $\boldsymbol{M}$ contains m regressor variables for each of the f levels of the factor. Direct fitting of the regression effects is facilitated by using the my basis function (`mbf` function) associating the regressor variables to the levels of the factor, essentially fitting $\boldsymbol{ZMv}$ where $\boldsymbol{Z}$ is the design matrix linking observations to the levels of the factor. But if m is much bigger than f, it is more computational efficient to fit an equivalent model $\boldsymbol{Zu}$ with a variance structure for $\boldsymbol{u}$ based on $\boldsymbol{MM}'$. ASReml can read the matrix $\boldsymbol{M}$ associated with a factor and group of regressor variables from a `.grr` file, construct a GRM matrix ($\boldsymbol{G} = \boldsymbol{MM'}/s$), fit the equivalent model and report both factor and regressor predictions. One common case of this model is when $\boldsymbol{u}$ represents genotype effects, the regressors represent SNP marker counts (typically 0/1/2) and $\boldsymbol{v}$ are marker effects.

The `.grr` file is specified after any pedigree file and before the data file (with any other GRM files). There may only be one `.grr` file. It is assumed to contain a row for each level of the factor, each row containing m regressor values. Optionally the factor level name associated with the i-th row can be included before the relevant regressor values. Also a heading row might include a name for each field/regressor variable. Superfluous fields before the factor or regressor fields can be skipped and superfluous rows before the regressor information can be skipped.

The syntax for specifying and reading the `.grr` file is

M.grr [!CSKIP c_1] ***Factor*** [f] [!NOID] [!CSKIP c_2] ***Regressors*** [m] [!NONAMES] [!SKIP s] where

M.grr is the name of the file to be read, !CSKIP c_1 indicates c_1 fields are to be skipped before the factor identifiers are read,

Factor is the name of the variable in the data that is associated with the regressors,

f sets the maximum number of levels (default 1000) of *Factor* with regressor data; ASReml will count the actual number,

!NOID indicates that the factor identifiers are not present in the .grr file,

!CSKIP c_2 indicates c_2 fields are to be skipped before the regressor variables are read,

Regressors is the name for the set of regressor variables,

m sets the number of regressor variables (default is the number of names found); must be set if there are extraneous fields to be ignored,

!SKIP s specifies how many lines are to be skipped before reading the regressor data,

!NONAMES indicates there is no line containing the individual names of the regressor variables; otherwise names are taken from the first (non-skipped) line in the file.

If the factor identifiers are not present (!NOID), ASReml assumes that the order of the factor classes in the data file matches the order in the .grr file. If the factor identifiers are present, ASReml uses the identifiers obtained from the .grr file to define the order of the factor classes when the data is read; any extra identifiers in the data not in the .grr file are appended at the end of the factor level name list. If !NOID is set, identifiers in the .grr file are not needed and if present should be skipped using !CSKIP.

Values are typically TAB, COMMA or SPACE separated but may be packed (no separator) when all values are integers 0/1/2. Missing values in the regression variables may be represented by *, NA. Invalid data is also treated as missing. Missing values are replaced by the mean of the respective regressor. Alternative missing data methods that involve imputation from neighbouring markers have not been implemented.

Some general qualifiers are:
!SAVEGIV instructs ASReml to write the $\boldsymbol{G}$ matrix in .dgiv format,
!PSD s declares that the derived variance matrix may have up to s singularities,
!PEV requests calculation of Prediction Error Variance of marker effects which are reported in the .mef file. Calculation of Prediction error variances is computationally very expensive,
!CENTRE [c] requests ASReml to centre the regressors at c if c is specified else at the individual regressor means; otherwise the $\boldsymbol{G}$ matrix is formed from uncentered regressors. Note that centring introduces a singularity in the G matrix and !PSV s will need to be set.

Other qualifiers relate specifically to whether the regressors are markers. Markers are typically coded 0/1/2 being counts of the minor allele. However, if they are imputed, they will take real values between 0 and 2. Since marker files may be huge,

`!SMODE` b sets the storage mode for the regressor data, indicating whether it is marker data: $b = 2$ sets 2bit storage for strictly 0/1/2 marker data, $b = 8$ (the default) sets 8bit storage useful for marker data with imputed values having 2 digits after the decimal, $b = 16$ sets 16bit storage useful for marker data with imputation with more than 2 digits and $b = 32$ sets 32bit real storage and should be used for non-marker data,
`!RANGE` l h indicates the marker scores range $l : h$ and are to be transformed to have a range 0:2,
`!GSCALE` s, controls the scaling of the GRM matrix. If unspecified $s = \Sigma 2p(1-p)$ is used for marker data, $s = 1$ for non marker data (`!SMODE 32`). Scaling is often used with centred marker data to scale the $\boldsymbol{MM}'$ matrix so that it is a genomic matrix.

Example

```
 !WORK 1
Nassau Clone Data
 Nfam 71 !A
 Nfemale 26 !A
 Nmale 37 !A
 Clone !A 860
 rep 8
 iblk 80
 culture !A
 DBH6

snpData.grr Clone Marker

nassau.csv !MAXIT 30 !SKIP 1 !DFF -1
DBH6 ~ mu culture/rep !r grm1v(Clon) 0.27 Clone 0.15 rep.iblk 0.31
```

where `snpData.grr` is first used to declare Clone identifiers (taken from the first field) in the correct order, and then contains the marker scores; it looks like

```
Genotype,0-10024-01-114,0-10037-01-257,0-10040-02-394,...
140099,2,2,1,2,2,2,2,2,2,1,2,1,2,1,1,2,1,2,2,2,2,2,1,2...
141099,2,2,0,0,2,2,1,2,2,1,2,1,2,2,0,2,2,2,2,1,2,2,1,1...
...
547853,2,2,1,2,2,2,1,2,2,0,2,1,2,2,2,2,2,2,2,1,2,...
547966,2,2,1,1,1,2,0,2,2,1,2,2,2,2,2,2,2,2,2,1,2,...
548082,2,2,1,2,2,2,1,2,1,2,2,1,2,2,1,2,2,2,2,1,2,...
```

The primary output follows.

```
 Nfam 71 !A
 Nfemale 26 !A
 Nmale 37 !A
 Clone  !A 860
 MatOrder 914 !A
 rep 8 !A
 iblk 80 !A
 prop 1 !A
 culture 2 !A
 treat 2 !A
 measure 1 !A
 CWAC6 !M-9
 Parsing: snpData.grr Clone
 Class names for factor "Clone" are initialized from the .grr file.
 GRR Header line begins: Genotype,0-10024-01-114,0-10037-01-257,0
        4854 Marker labels found
Marker labels 0-10024-01-114 ... UMN-CL98Contig1-
 Notice: The header line indicates there are 4854 regressors in the file.
 Notice: SNP data line begins: 140099,2,2,1,2,2,2,2,2,2,1,2,1,2,1,1,
 Notice: Markers coded -9 treated as missing.
 Marker data [0/1/2] for 923 genotypes and 4854 markers read from snpData.grr
      160414 missing Regressor values (  3.6%) replaced by column average!
        Regressor values ranged 0.00 to 2.00
        Regressor Means ranged  1.00 to 2.00
          Sigma2p(1-p) is   1057.12558
 GIV1  snpData.grr      923       9     -946.27
 QUALIFIERS: !MAXIT 30 !SKIP 1 !DFF -1
 QUALIFIER: !DOPART    2 is active
 Reading nassau_cut_v3.csv  FREE FORMAT skipping     1 lines

 Univariate analysis of HT6
 Summary of 6399 records retained of 6795 read

  Model term          Size #miss #zero   MinNon0    Mean      MaxNon0  StndDevn
   1 Nfam               71     0     0      1   36.3379         71
   2 Nfemale            26     0     0      1   12.8823         26
   3 Nmale              37     0     0      1   15.2285         37
  Warning: More levels found in Clone  than specified
   4 Clone             926     0     0      1   464.6765        926
  Warning: Fewer levels found in MatOrder  than specified
   5 MatOrder          914     0     0      1   432.5760        860
   6 rep                 8     0     0      1     4.4837          8
   7 iblk               80     0     0      1   40.1164         80
   8 tree                      0     0 1.0000      7.473      14.00      4.018
   9 row                       0     0 1.0000      28.52      56.00      16.09
  10 col                       0     0 1.0000      10.50      20.00      5.760
  Warning: Fewer levels found in prop  than specified
  11 prop                2     0     0      1     1.0000          1
  12 culture             2     0     0      1     1.4945          2
  13 treat               2     0     0      1     1.4945          2
  Warning: Fewer levels found in measure  than specified
  14 measure             2     0     0      1     1.0000          1
  15 SURV                      0     6 1.0000     0.9991     1.0000     0.3061E-01
  16 DBH6                      4     0 0.3000E-01  11.29      18.80      2.400
  17 HT6            Variate    0     0  76.20      838.6      1286.      163.6
  18 HT8                      83     0  91.44      1148.      1576.      170.6
  19 CWAC6                  3167     0  97.54      301.3      542.5      52.26
  20 mu                          1
  21 culture.rep                16 12 culture   :   2   6 rep            :    8
 Warning: GRM matrix is too SMALL
  22 grm1(Clone)       923
  23 rep.iblk                  640  6 rep       :   8   7 iblk           :   80
 Forming     2508 equations:  19 dense.
 Initial updates will be shrunk by factor    0.316
 Notice: LogL values are reported relative to a base of -30000.000
 Notice:     11 singularities detected in design matrix.
   1 LogL=-2845.97     S2=  8956.5       6390 df
   2 LogL=-2799.30     S2=  8568.1       6390 df
   3 LogL=-2759.03     S2=  8131.3       6390 df
   4 LogL=-2741.99     S2=  7766.2       6390 df
   5 LogL=-2741.40     S2=  7702.9       6390 df
   6 LogL=-2741.40     S2=  7700.1       6390 df

          - - - Results from analysis of HT6 - - -
 Akaike Information Criterion    65490.79 (assuming 4 parameters).
 Bayesian Information Criterion  65517.84

 Model_Term                             Gamma         Sigma   Sigma/SE   % C
 rep.iblk                IDV_V  640  0.307856       2370.52      13.00   0 P
 grm1(Clone)             GRM_V  923  0.275656       2122.58       5.82   0 P
 Clone                   IDV_V  926  0.152554       1174.68       6.08   0 P
 Residual                SCA_V 6399  1.000000       7700.10      49.64   0 P

                                   Wald F statistics
     Source of Variation           NumDF              F-inc
  20 mu                                1           0.11E+06
  12 culture                           1            2615.96
  21 culture.rep                       6              30.44
  23 rep.iblk                            640 effects fitted
  22 grm1(Clone)                         923 effects fitted
   4 Clone                               926 effects fitted (      66 are zero)
          78  possible outliers: see .res file
```

Notes:

- of 926 clones identified, 860 have data and 923 have genomic data.
- The `.res` file contains additional details about the analysis including a listing of the larger marker effects. All marker effects are reported in the `.mef` file.
- Particular columns of the `.grr` data can be included in the model using the `grr(`*`Factor`*`,`*`i`*`)` model term where and i specifies which (number) regressor variable to include.

```
Listing of the larger marker effects
      368  0-12761-01-121     1.40736       0.00000
      617  0-14383-01-111     1.26081       0.00000
      777  0-15417-01-138    -1.25597       0.00000
     1246  0-18644-02-210     1.22522       0.00000
     1903  0-6963-01-202     -1.24800       0.00000
     2102  0-8683-02-432      1.15496       0.00000
     2445  2-1563-02-244     -1.35181       0.00000
```

```
2497  2-2167-01-413     -1.21339       0.00000
3180  2-8668-03-42      -1.21629       0.00000
3521  CL1577Contig1-03  -1.15833       0.00000
3802  CL2573Contig1-03   1.17005       0.00000
4195  CL595Contig1-01-  -1.19330       0.00000
4351  UMN-1397-01-416   -1.34916       0.00000
```

10 Tabulation of the data and prediction from the model

10.1 Introduction

This chapter describes the `tabulate` directive and the `predict` directive introduced in Section 3.4 under Prediction.

Tabulation is the process of forming simple tables of averages and counts from the data. Such tables are useful for looking at the structure of the data and numbers of observations associated with factor combinations. Multiple tabulate directives may be specified in a job.

Prediction is the process of forming a linear function of the vector of fixed and random effects in the linear model to obtain an estimated or predicted value for a quantity of interest. It is primarily used for predicting tables of adjusted means. If a 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.

10.2 Tabulation

A `tabulate` directive is provided to enable simple summaries of the data to be formed for the purpose of checking the structure of the data. The summaries are based on the same records as are used in the analysis of the model fitted in the same run. In particular, it will ignore records that exist in the data file but were dropped as the data was read into ASReml, either explicitly using `!DV` or implicitly because the dependent variable had missing values. Multiple `tabulate` statements are permitted either immediately before or after the linear model. If a linear (mixed) model is not supplied, tabulation is based on all records.

The tabulate statement has the form

`tabulate` *response_variables* [`!WT` *weight* `!COUNT !DECIMALS [`*d*`] !SD !RANGE !STATS !FILTER` *filter* `!SELECT` *value*] ~ *factors*

- `tabulate` is the directive name, appearing on a new line,
- *response_variables* is a list of variates for which means are required,
- `!WT` *weight* nominates a variable containing weights,
- `!COUNT` requests counts as well as means to be reported,
- `!DECIMALS` [d] ($1 \leq d \leq 7$) requests means be reported with d decimal places. If omitted, ASReml reports 5 significant digits; if specified without an argument, 2 decimal places are reported,
- `!RANGE` requests the minimum and maximum of each cell be reported,
- `!SD` requests the standard deviation within each cell be reported,
- `!STATS` is shorthand for `!COUNT !SD !RANGE`,
- `!FILTER` *filter* nominates a factor for selecting a portion of the data,
- `!SELECT` *value* indicates that only records with *value* in the *filter* column are to be included,
- $\sim$ *factors* identifies the factors to be used for classifying the data. Only factors (not covariates) may be nominated and no more than six may be nominated.

ASReml prints the multiway table of means omitting empty cells to a file with extension `.tab`.

10.3 Prediction

10.3.1 Underlying principles

Our approach to prediction is a generalization of that of Lane and Nelder (1982) who only consider fixed effects models. They form fitted values for all combinations of the explanatory variables in the model, then take marginal means across the explanatory variables not relevent to the current prediction. Our case is more general in that we also consider the case of associated factors (see below) and options for random effects that appear in our (mixed) models. A formal description can be found in Gilmour *et al.* (2004) and Welham *et al.* (2004).

Associated factors have a particular one to many association such that the levels of one factor (say Region) define groups of the levels of another factor (say Location). In prediction, it is necessary to correctly associate the levels of associated factors.

10.3 Prediction

Terms in the model may be fitted as fixed or random, and are formed from explanatory variables which are either factors or covariates. For this exposition, we define a *fixed factor* as an explanatory variable which is a factor and appears in the model in terms that are fixed (it may also appear in random terms), a *random* factor as an explanatory variable which is a factor and appears in the model only in terms that are fitted as random. Covariates generally appear in fixed terms but may appear in random terms as well (random regression). In special cases they may appear only in random terms.

Random factors may contribute to predictions in several ways. They may be evaluated at levels specified by the user, they may be averaged over, or they may be ignored (omitting all model terms that involve the factor from the prediction). Averaging over the set of random effects gives a prediction specific to the random effects observed. We call this a 'conditional' prediction. Omitting the term from the prediction model produces a prediction at the population average (often zero), that is, substituting the assumed population mean for an predicted random effect. We call this a 'marginal' prediction. Note that in any prediction, some random factors (for example Genotype) may be evaluated as conditional and others (for example Blocks) 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. Therefore any prediction will be either for specific levels of the fixed factor, or averaging (in some way) over the levels of the fixed factor. The prediction will therefore involve all fixed model terms.

Covariates must be predicted at specified values. If interest lies in the relationship of the response variable to the covariate, predict a suitable grid of covariate values to reveal the relationship. Otherwise, predict at an average or typical value of the covariate. The default is to predict at the mean covariate value. Omission of a covariate from the prediction model is equivalent to predicting at a zero covariate value, which is often not appropriate (unless the covariate is centred).

Before considering the syntax, it is useful to consider the conceptual steps involved in the prediction process. Given the explanatory variables (fixed factors, random factors and covariates) used to define the linear (mixed) model, the four main steps are

(a) Choose the explanatory variable(s) and their respective level(s)/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. Include only one from any set of associated factors in the classify set.

(b) Note which of the remaining variables will be averaged over, the *averaging* set, and which will be ignored, the *ignored* set. The *averaging* set will include all variables involved in the fixed model but not in the classify set. Ignored variables may be explicitly added to the averaging set. The combination of the classify set with these averaging variables defines a multiway hyper-table. Only the base factor in a set of associated factors formally appears in this hyper-table, regardless of whether it is fitted as fixed or random. Note that variables evaluated at only one value, for example, a covariate at its mean value, can be formally introduced as part of the classify or averaging set.

(c) Determine which terms from the linear mixed model are to be used when predicting the cells in the multiway hyper-table in order to obtain either conditional or marginal predictions. That is, you may choose to ignore some random terms in addition to those ignored because they involve variables in the ignored set. All terms involving associated factors are by default included.

(d) Choose the weights to be used when averaging cells in the hyper-table to produce the multiway table to be reported. The multiway table may require partial and/or sequential averaging over associated factors. Operationally, ASReml does the averaging in the prediction design matrix rather than actually predicting the cells of the hyper-table and then averaging them.

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, factors not in the classify set must be in the averaging set, and all terms must contribute to the predictions.

10.3.2 Predict syntax

The first step is to specify the classify set of explanatory variables after the `predict` directive. The `predict` statement(s) may appear immediately after the model line (before or after any `tabulate` statements) or after the R and G structure lines. The syntax is

```
NIN Alliance trial 1989  variety !A
:
 column 11
nin89.asd !skip 1
yield ~ mu variety !r idv(repl)
predict variety
```

`predict` *factors* [*qualifiers*]

- `predict` must be the first element of the `predict` statement, in upper or lower case,
- *factors* is a list of the variables defining a multiway table to be predicted; each variable may be followed by a list of specific levels/values to be predicted, or the name of the file that contains those values,
- the *qualifiers*, listed in Table 10.1, modify the predictions in some way,
- a `predict` statement may be continued on subsequent lines by terminating the current line with a comma,
- several `predict` statements may be specified.

ASReml parses each `predict` statement before fitting the model. If any syntax problems are encountered, these are reported in the `.pvs` file after which the statement is ignored: the job is completed as if the erroneous prediction statement did not exist.

The predictions are formed as an extra process in the final iteration and are reported to the .pvs file. Consequently, aborting a run by creating the ABORTASR.NOW file (see page 68) will cause any predict statements to be ignored. Create FINALASR.NOW instead of ABORTASR.NOW to make the next iteration, the final iteration in which prediction is performed.

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 !G qualifier page 48) are treated as covariates.

Prediction at particular values of a covariate or particular levels of a factor is achieved by listing the levels/values after the variate/factor name. Where there is a sequence of values, use the notation a b ... n to represent the sequence of values from a to n with step size $b-a$. The default stepsize is 1 (in which case b may be omitted). A colon (:) may replace the ellipsis (...). An increasing sequence is assumed. When giving particular values for factors, the default is to use the coded level (1:n) rather than the label (alphabetical or integer). To use the label, precede it with a quote ("). Where a large number of values must be given, they can be supplied in a separate file, and the filename specified in quotes. The file form does not allow label coding or sequences. (See the discussion of !PRWTS for an example.)

Model terms mv and units are always ignored.

Model terms which are functions (such as at(, and(, pol(, sin(, spl(, ...) including those defined using !CONTRAST, !GROUP, !SUBGROUP, !SUBSET and !MBF are implicitly defined through their base variables and can not be directly referenced in the classify and average sets. For example,

```
!GROUP Year YearLoc 1 1 1 2 2 3 3 3 4 4
```

forms a new factor Year with 4 levels from the existing factor YearLoc with 10 levels. The prediction must be in terms of YearLoc, not Year even if YearLoc does not formally appear in the model. For default averaging in prediction, the weights for the levels of the grouped factor (Year) will be (in this example) 0.3 0.2 0.3 0.2 derived from the weights for the base factor (YearLoc). Use !AVE YearLoc { 2 2 2 3 3 2 2 2 3 3 }/24 to produce equal weighting of Year effects.

If !G sets of variables are included in the classify set, only the first variable is reported in labelling the predict values, except that for !G !MM sets, the marker position is reported.

Having identified the explanatory variables in the classify set, the second step is to check the averaging set. The default averaging set is those explanatory variables involved in fixed effect model terms that are not in the classify set. By default variables that are not in any !ASSOCIATE list and that only define random model terms are ignored. Use the !AVERAGE, !ASSOCIATE or !PRESENT, qualifiers to force variables into the averaging set.

The third step is to check the linear model terms to use in prediction. The default is that all model terms based entirely on variables in the classifying and averaging sets are used. Two qualifiers allow this default to be modified by adding (!USE) or removing (!IGNORE) model terms. The qualifier !ONLYUSE explicitly specifies the model terms to use, ignoring all others. The qualifier !EXCEPT explicitly specifies the model terms not to use, including all others. These qualifiers will not override the definition of the averaging set.

The fourth step is to choose the weights to use when averaging over dimensions in the hypertable. The default is to simply average over the specified levels but the qualifier !AVERAGE *factor weights* allows other weights to be specified. !PRESENT and !ASSOCIATE/!ASAVERAGE generate more complicated averaging processes.

The basic prediction process is described in the following example:

```
yield ~ site variety !r idv(site).id(variety) at(site).idv(block)
predict variety
```

puts variety in the classify set, site in the averaging set and block in the ignore set. Consequently, ASReml implicitly forms the site×variety hyper-table from model terms site, variety and site.variety but ignoring all terms in at(site).block, and then averages across the sites to produce variety predictions. This prediction will work even if some varieties were not grown at some sites because the site.variety term was fitted as random. If site.variety was fitted as fixed, variety predictions would be non estimable for those varieties that were not grown at every site.

10.3.3 Predict failure

It is not uncommon for users to get the message

```
Warning:  non-estimable [aliased] cell(s) may be omitted.
```

because ASReml checks that predictions are of estimable functions in the sense defined by Searle (1971, p160) and are invariant to any constraint method used.

Immediate things to check include whether every level of every fixed factor in the averaging set is present, and whether all cells in every fixed interaction is filled. For example, in the previous example, no variety predictions would be obtained if `site` was declared as having 4 levels but only three were present in the data. The message is also likely if any fixed model terms are `!IGNORE`d. The `TABULATE` command may be used to see which treatment combinations occur and in what order.

More formally, there are often situations in which the fixed effects design matrix $\boldsymbol{X}$ is not of full column rank. This aliasing has three main causes.

- linear dependencies among the model terms due to over-parameterisation of the model,
- no data present for some factor combinations so that the corresponding effects cannot be estimated,
- linear dependencies due to other, usually unexpected, structure in the data.

The first type of aliasing is imposed by the parameterisation chosen and can be determined from the model. The second type of aliasing can be detected when setting up the design matrix for parameter estimation (which may require revision of imposed constraints). All types are detected in ASReml during the absorption process used to obtain the predicted values.

ASReml doesn't print predictions of non-estimable functions unless the `!PRINTALL` qualifier is specified. However, using `!PRINTALL` is rarely a satisfactory solution. Failure to report predicted values normally means that the `predict` statement 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` and/or `!PRESENT` qualifiers will usually resolve the problem. The `!PRESENT` qualifier enables the construction of means by averaging only the estimable cells of the hyper-table, where this is appropriate.

Table 10.1 is a list of the prediction qualifiers with the following syntax:

- f is an explanatory variable which is a factor,
- t is a list of terms in the fitted model,
- n is an integer number,
- v is a list of explanatory variables.

10.3 Prediction

Table 10.1: List of prediction qualifiers

qualifier	action
Controlling formation of tables	
`!ASSOCIATE` [*v*]	facilitates prediction when the levels of one factor are grouped by the levels of another in a hierarchical manner. More details are given below. Two independent associate lists may be specified.
`!AVERAGE` *f* [*weights*] `!AVERAGE` *f* '*file*'[,*n*]	is used to formally include a variable in the averaging set and to explicitly set the weights for averaging. Variables that only appear in random model terms are not included in the averaging set unless specified with the `!AVERAGE`, `!ASSOCIATE` or `!PRESENT` qualifiers. Explicit weights may be supplied directly or from a file. The default is equal weights. *weights* can be expressed like `{3*1 0 2*1}/5` to represent the sequence `0.2 0.2 0.2 0 0.2 0.2`. The string inside the curly brace is expanded first and the expression $n*c$ means n occurrences of c. When there are a large number of weights, it may be convenient to prepare them in a file and retrieve them. All values in the file are taken unless ',*n*' is specified in which case they are taken from field/column *n*.
`!ASAVERAGE` *f* [*weights*] `!ASAVERAGE` *f* '*file*'[,*n*]	is used to control averaging over associated factors. The default is to simply average at the base level. Hierarchal averaging is achieved by listing the associated factors to average in *f*. Explicit weights may be supplied directly or from a file as for `!AVERAGE`.
`!PARALLEL` [*v*]	without arguments means all classify variables are expanded in parallel. Otherwise list the variables from the classify set whose levels are to be taken in parallel.
`!PRESENT` *v*	is used when averaging is to be based only on cells with data. *v* is a list of variables and may include variables in the classify set. *v* may not include variables with an explicit `!AVERAGE` qualifier. The variable names in *v* may optionally be followed by a list of levels for inclusion if such a list has not been supplied in the specification of the classify set. ASReml works out what combinations are present from the design matrix. It may have trouble with complicated models such as those involving `and()` terms. A second `!PRESENT` qualifier is allowed on a `predict` statement (but not with `!PRWTS`). The two lists must not overlap.
`!PRWTS` *v*	is used in conjunction with the first `!PRESENT` *v* list to specify the weights that ASReml will use for averaging that `!PRESENT` table. More details are given below.

Table 10.1: List of prediction qualifiers

qualifier	action
Controlling inclusion of model terms	
`!EXCEPT` t	causes the prediction to include all fitted model terms not in t.

`!IGNORE` t	causes ASReml to set up a prediction model based on the default rules and then removes the terms in t. This might be used to omit the spline Lack of fit term (`!IGNORE fac(x)`) from predictions as in
	`yield ~ mu x variety !r spl(x) fac(x)`
	`predict x !IGNORE fac(x)`
	which would predict points on the spline curve averaging over `variety`.

`!ONLYUSE` t	causes the prediction to include only model terms in t. It can be used for example to form a table of slopes as in
	`HI ~ mu X variety X.variety`
	`predict variety X 1 !onlyuse X X.variety`

`!USE` t	causes ASReml to set up a prediction model based on the default rules and then adds the terms listed in t.
Printing	
`!DEC` [n]	gives the user control of the number of decimal places reported in the table of predicted values where n is 0...9. The default is 4. `G15.9` format is used if n exceeds 9. When `!VVP` or `!SED` are used, the values are displayed with 6 significant digits unless n is specified and even; then the values are displayed with 9 significant digits.
`!PLOT` [x]	instructs ASReml to attempt a plot of the predicted values. This qualifier is only applicable in versions of ASReml linked with the Winteracter Graphics library. If there is no argument, ASReml produces a figure of the predicted values as best it can. The user can modify the appearance by typing <Esc> to expose a menu or with the plot arguments listed in Table 10.2.
`!PRINTALL`	instructs ASReml to print the predicted value, even if it is not of an estimable function. By default, ASReml only prints predictions that are of estimable functions.
`!SED`	requests all *standard errors of difference* be printed. Normally only an average value is printed. Note that the default average SED is actually an SED calculated from the average variance if the predicted values and the average covariance among the predicted values rather than being the average of the individual SED values. However, when `!SED` is specified, the average of the individual SED values is reported.
`!TDIFF`	requests t-statistics be printed for all combinations of predicted values.
`!TURNINGPOINTS` n	requests ASReml to scan the predicted values from a fitted line for possible turning points and if found, report them and save them internally in a vector which can be accessed by subsequent parts of the same job using `$TP`n. This was added to facilitate location of putative QTL (Gilmour, 2007).

Table 10.1: List of prediction qualifiers

qualifier	action
`!TWOSTAGEWEIGHTS`	is intended for use with variety trials which will subsequently be combined in a meta analysis. It forms the variance matrix for the predictions, inverts it and writes the predicted variety means with the corresponding diagonal elements of this matrix to the `.pvs` file. These values are used in some variety testing programs in Australia for a subsequent second stage analysis across many trials (Smith *et al.*, 2001). A data base is used to collect the results from the individual trials and write out the combined data set. The diagonal elements, scaled by the variance which is also reported and held in the data base, are used as weights in the combined analysis.
`!VPV`	requests that the variance matrix of predicted values be printed to the `.pvs` file.

PLOT graphic control qualifiers

This functionality was developed and this section was written by Damian Collins.

The `!PLOT` qualifier produces a graphic of the predictions. Where there is more than one prediction factor, a multi-panel 'trellis' arrangement may be used. Alternatively, one or more factors can be superimposed on the one panel. The data can be added to the plot to assist informal examination of the model fit.

With no plot options, ASReml chooses an arrangement for plotting the predictions by recognising any covariates and noting the size of factors. However, the user is able to customize how the predictions are plotted by either using options to the `!PLOT` qualifier or by using the graphical interface. The graphical interface is accessed by typing Esc when the figure is displayed.

The `!PLOT` qualifier has the following options:

Table 10.2: List of predict plot options

option	action
Lines and data	
`^addData`	superimposes the raw data.

10.3 Prediction

Table 10.2: List of predict plot options

option	action
`ˆaddlabels` *factors*	superimposes the raw data with the data points labelled using the given factors (which must not be prediction factors). This option may be useful to identify individual data points on the graph – for instance, potential outliers – or alternatively, to identify groups of data points (e.g. all data points in the same stratum).
`ˆaddlines` *factors*	superimposes the raw data with the data points joined using the given factors which must not be prediction factors. This option may be useful for repeated measures data.
`ˆnoSEs`	specifies that no error bars should be plotted (by default, they are plotted)
`ˆsemult` *r*	specifies the multiplier of the SE used for creating error bars (default=1.0)
`ˆjoinmeans`	specifies that the predicted values should be joined by lines (by default, they are only joined if the x-axis variable is numeric)
Predictions involving two or more factors	
	If these arguments are used, all prediction factors (except for those specified with only one prediction level) must be listed once and only once, otherwise these arguments are ignored.
`ˆxaxis` *factor*	specifies the prediction factor to be plotted on the x-axis
`ˆsuperimpose` *factors*	specifies the prediction factors to be superimposed on the one panel.
`ˆcondition` *factors*	specifies the conditioning factors which define the panels. These should be listed in the order that they will be used.
Layout	
`ˆgoto` *n*	specifies the page to start at, for multi-page predictions.
`ˆsaveplot` *filename*	specifies the name of the file to save the plot to.
`ˆlayout` *rows cols*	specifies the panel layout on each page
`ˆbycols`	specifies that the panels be arranged by columns (default is by rows)
`ˆblankpanels` *n*	specifies that each page contains n blank panels. This sub-option can only be used in combination with the layout sub-option.
`ˆextrablanks` *n* and `ˆextraspan` *p*	specifies that an additional *n* blank panels be used every *p* pages These can only be used with the `layout` sub-option.
Improving the graphical appearance (and readability)	
`ˆlabcharsize` *n*	specifies the relative size of the data points/labels (default=0.4)
`ˆpanelcharsize` *n*	specifies the relative size of the labels used for the panels (default=1.0)
`ˆvertxlab`	specifies that vertical annotation be used on the x-axis (default is horizontal).
`ˆabbrdlab` *n*	specifies that the labels used for the data be abbreviated to *n* characters.
`ˆabbrxlab` *n*	specifies that the labels used for the x-axis annotation be appreviated to *n* characters.

Table 10.2: List of predict plot options

option	action
ˆ`abbrslab` *n*	specifies that the labels used for superimposed factors be abbreviated to *n* characters.

10.3.4 Associated factors

`!ASSOCIATE` *factors* facilitates prediction when the levels of one factor group or classify the levels of another, especially when there are many levels. *factors* is the list of factors in the model which have this hierarchical relationship. Typical examples are individually named lines grouped into families, usually with unequal numbers of lines per family, or trials conducted at locations within regions.

Declaring factors as associated allows ASReml to combine the levels of the factors appropriately. For example, when predicting a trial mean, to add the effect of the location and region where the trial was conducted. When identifying which levels are associated, ASReml checks that the association is strictly hierarchal, tree-like. That is, each trial is associated with one location and each location is associated with only one region. If a level code is missing for one component, it must be missing for all.

Averaging of associated factors will generally give differing results depending on the order in which the averaging is performed. We explore this with the following extended example. Consider the mean yields from 15 trials classified by region and location in Table 10.4.

Table 10.3: Trials classified by region and location

	location							
Region	L1	L2	L3	L4	L5	L6	L7	L8
R1	T1, T2	T3, T4, T5	T6					
R2				T7, T8	T9, T10, T11	T12, T13	T14	T15

Table 10.4: Trial means

T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11	T12	T13	T14	T15
10	12	11	12	13	13	11	13	11	12	13	10	12	10	10

Assuming a simplified linear model `yield ~ mu region location trial`
the predict statement `predict trial !ASSOCIATE region location trial`
will reconstruct the 15 trial means from the fitted mu, region, location and trial effects.

Given these trial means, it is fairly natural to form location means by averaging the trials in each location to get the location means in Table 10.5.

Table 10.5: Location means

L1	L2	L3	L4	L5	L6	L7	L8
11	12	13	12	12	11	10	10

These are given by

```
predict location !ASSOCIATE region location trial !ASAVERAGE trial
```

or equivalently

```
predict location !ASSOCIATE region location trial
```

since the default is to average the base associate factor (trial) within the associated classify factor (location).

By contrast, by specifying
`predict location`
or equivalently
`predict location !AVERAGE region !AVERAGE trial`
ASReml would add the average of all the trial effects and the average of the region effects into all of the location means which is not appropriate. With `!ASSOCIATE`, it knows which trials to average (and which region effects to include) to form each location mean. That is, ASReml knows how to construct the trial means including the appropriate region and location effects, and which trials means to then average to form the location table.

However, for region means, we have a choice. We can average the trial means in Table 10.4 according to region obtaining region means of 11.83 and 11.33, or we can average the location means in Table 10.5 to get region means of 12 and 11.

The former is the default in ASReml produced by
`predict region !ASSOCIATE region location trial !ASAVERAGE trial`
or equivalently by
`predict region !ASSOCIATE region location trial`
Again, this is *base* averaging.

By contrast,
`predict region !ASSOC region location trial !ASAVE location trial`
(or `predict region !ASSOC region location trial !ASAVE location`)
produces sequential averaging giving region means of 12 and 11 respectively.

Similarly, an overall sequential mean of 11.5 is given by
`predict mu !ASSOC region location trial !ASAVE region location`
while `predict mu !ASSOC region location trial !ASAVE region`
gives a value of 11.58 being the average of region means 11.83 and 11.33 obtained by averaging trials within regions from Table 10.4, and
`predict mu !ASSOCIATE region location trial !ASAVE location`
predicts mu as 11.38, the average of the 8 location means in Table 10.5.

Further discussion of associated factors

The user may specify their own weights, using file input if necessary. Thus `predict region ... !ASAVERAGE location {1 2 3}/6 {1 1 1 2 1}/6` would give region predictions of 11.67 and 10.84 respectively derived from the location predictions in Table 10.5. Note that because location is nested in region, the location weights should sum to 1.0 within levels of region when forming region means. The `!AVERAGE` (`!ASAVERAGE`) qualifier allows the weights to be read from a file which the user can create elsewhere. Thus the code `!ASAVERAGE trial 'Tweight.csv',2` will read the weights from the second field of file `Tweight.csv`. The user must ensure the weights are in the coding order ASReml uses (`trial` order in this instance, given in the `.sln` file or by using the `TABULATE` command).

It was noted that it is the base `!ASSOCIATE` factor that is formally included in the hypertable. If the lowest stratum is random, it may be appropriate to ignore it. Omitting it from the !ASSOCIATE list will allow it to reenter the Ignore set. Specifying it with the !IGNORE qualifier will exclude its effects from the prediction but not ignore the structural information implied by the association.

Normally it is not necessary for any model term to involve more than 1 of the associated factors. One exception is if an interaction is required so that the variance can differ between sections. For example, fitting the terms `at(region).trial` as random effects would allow the trials in region 1 to have a different variance component to those in region 2. Prediction in these cases is more complicated and has only been implemented for this specific case and the analagous `region.trial` case. The associated factors must occur together in this order for the prediction to give correct answers.

The !ASSOCIATE effect (with base averaging) can usually be achieved with the !PRESENT qualifier except when the factors have many levels so that the product of levels exceeds 2147 000 000; it fails in this case because the KEY for identifying the cells present is a simple combination of the levels and is stored as a normal (32bit) integer. However, !ASSOCIATE is preferred because it formally checks the association structure as well as allowing sequential averaging.

Two !ASSOCIATE clauses may be specified for example

```
PRED entry !ASSOC family entry !ASSOC reg loc trial !ASAVE reg loc.
```

Only one member of an !ASSOCIATE list may also appear in a !PRESENT list. If one member appears in the classify set, only that member may appear in the !PRESENT list. For example

```
yield ~ region !r idv(region).id(family) idv(entry)
PREDICT entry !ASSOCIATE family entry !PRESENT entry region .
```

Association averaging is used to form the cells in the PRESENT table and PRESENT averaging is then applied.

10.3.5 Complicated weighting with !PRESENT

Generally, when forming a prediction table, it is necessary to average over (or ignore) some dimensions of the hyper table. By default, ASReml uses equal weights ($1/f$ for a factor with f levels). More complicated weighting is achieved by using the !AVERAGE qualifier to set specific (unequal) weights for each level of a factor. However, sometimes the weights need to be defined with respect to two or more factors. The simplest case is when there are missing cells and weighting is equal for those cells in a multiway table that are present; achieved by using the !PRESENT qualifier. This is further generalized by allowing the user to supply the weights to be used by the !PRESENT machinery via the !PRWTS qualifier.

The user specifies the factors in the table of weights with the !PRESENT statement and then gives the table of weights using the !PRWTS qualifier. There may only be one !PRESENT qualifier on the `predict` line when !PRWTS is specified. 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. The weights may be given in a separate file if a filename (in quotes) is given as the argument to !PRWTS. Check the output to ensure that the values in the tables of weights are applied in the correct order. ASReml may transpose the table of weights to match the order it needs for processing.

When weights are supplied in a separate file, two layouts are allowed. The default is to read all values in the file, regardless of layout. Otherwise, the weights must appear a single column/field (one weight per line) where the field is specified by appending `,`*c* to the filename.

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

```
predict year 1 crop 1 pasture lime !AVE month 56 55 56 53 57 63 6*0
predict year 2 crop 1 pasture lime !AVE month 36 0 0 53 23 24 54 54 43 35 0 0
predict year 3 crop 1 pasture lime !AVE month 70 0 21 17 0 0 0 70 0 0 53 0
predict year 4 crop 1 pasture lime !AVE month 53 56 22 92 19 44 0 0 36 0 0 49
predict year 5 crop 1 pasture lime !AVE month 0 22 0 53 70 22 0 51 16 51 0 0
```

but to average over years as well, we need one of the following `predict` statements:

```
predict crop 1 pasture lime !PRES year month ,
 !PRWTS { 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
predict crop 1 pasture lime !PRES month year ,
 !PRWTS { 56 36 70 53 0,
          55  0  0 56 22,
          56  0 21 22  0,
          53 53 17 92 53,
          57 23  0 19 70,
          63 24  0 44 22,
           0 54  0  0  0,
           0 54 70  0 51,
           0 43  0 36 16,
           0 35  0  0 51,
           0  0 53  0  0,
           0  0  0 49  0}/5
predict crop 1 pasture lime !PRES year month  !PRWTS 'YMprwts.txt'
```

where `YMprwts.txt` contains

```
11.2 11.0 11.2 10.6 11.4 12.6  0.0  0.0 0.0  0.0  0.0  0.0
 7.2  0.0  0.0 10.6  4.6  4.8 10.8 10.8 8.6  7.0  0.0  0.0
14.   0.0  4.2  3.4  0.0  0.0  0.0 14.  0.0  0.0 10.6  0.0
10.6 11.2  4.4 18.4  3.8  8.8  0     0  7.2  0    0    9.8
```

```
0     4.4   0    10.6 14    4.4   0    10.2 3.2  10.2 0    0
```

We have presented both sets of predict statements to show how the weights were derived and presented. Notice that the order in !PRESENT year month implies that the weight coefficients are presented in standard order with the levels for months cycling within levels for years. There is a check which reports if non zero weights are associated with cells that have no data. The weights are reported in the .pvs file. !PRESENT counts are reported in the .res file.

10.3.6 Examples

Examples are as follows:

```
yield ∼ mu variety !r idv(repl)
predict variety
```

is used to predict variety means in the NIN field trial analysis. Random repl is ignored in the prediction.

```
yield ∼ mu x variety !r idv(repl)
predict variety
```

predicts variety means at the average of x ignoring random repl.

```
yield ∼ mu x variety repl
predict variety x 2
```

forms the hyper-table based on variety and repl at the covariate value of 2 and then averages across repl to produce variety predictions.

```
GFW Fdiam ∼ Trait Trait.Year !r idv(Trait).id(Team)
predict Trait Team
```

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

```
yield ∼ variety !r idv(site).id(variety)
predict variety
```

will ignore the site.variety term in forming the predictions while

```
predict variety !AVERAGE site
```

forms the hyper-table based on site and variety with each linear combination in each cell using variety and site.variety effects and then forms averages across sites to produce variety predictions.

```
yield ∼ site variety !r idv(site).id(variety) at(site).idv(block)
predict variety
```

puts variety in the classify set, site in the averaging set and block in the ignore set. Consequently, it forms the site×variety hyper-table from model terms site, variety and site.variety but ignoring all terms in at(site).block, and then forms averages across sites to produce variety predictions.

10.3.7 New R4 Prediction using two-way interaction effects

In some cases we wish to calculate from two way interaction effects, bc_{ij} say, effects for one of the factors, B say, that are a weighted sum averaged over the c levels of C, ie. $b_i = \sum_{j=1}^{c} bc_{ij} w_j$.

```
TPREDICT C !AVE B weights !ONLYUSE fun(B).fun(C)
```

allows this to be produced more computationally efficiently than it would be using `PREDICT`. For example,

```
TPREDICT Animal !AVE Trait 2.1 1.2 -7.4 !ONLYUSE us(Trait).nrm(Animal)
```

Part of the motivation for this is the calculation of selection indices. The index coefficients are typically derived as $w = a'G_{om}G_{mm}^{-1}$ where G_{mm} is the variance matrix for the measured traits (corresponding to C in the example), G_{om} is the genetic covariance matrix between the objective traits and the measured traits, and a is the vector of economic values for the objective traits. The results are given in a `.sli` (selection index) file. This directive should be placed after the model specification.

11 Command file: Running the job

11.1 Introduction

The command line, its options and arguments are discussed in this chapter. Command line options enable more workspace to be accessed to run the job, control some graphics output and control advanced processing options. Command line arguments are substituted into the job at run time.

As Windows likes to hide the command line, most command line options can be set on an optional initial line of the `.as` file we call *the top job control line* to distinguish it from the other job control lines discussed in Chapter 6. If the first line of the `.as` file contains a qualifier other than `!DOPATH`, it is interpreted as setting command line options and the *Title* is taken as the next line.

11.2 The command line

11.2.1 Normal run

The basic command to run ASReml is

[*path*]`ASReml` *basename*[`.as`[`c`]]

- *path* provides the path to the ASReml program (usually called `asreml.exe` in a PC environment). In a UNIX environment, ASReml is usually run through a shell script called `ASReml`.
  - if the ASReml program is in the search path then *path* is not required and the word `ASReml` will suffice; for example

    ```
    ASReml nin89.as
    ```

    will run the NIN analysis (assuming it is in the current working folder),

  - if `asreml.exe`(`ASReml`) is not in the search path then *path* is required, for example, if `asreml.exe` is in the usual place then

    ```
    C:\Program Files\ASReml3\bin\Asreml nin89.as
    ```

will run `nin89.as`,

- `ASReml` invokes the ASReml program,
- *basename* is the name of the `.as`[`c`] command file.

The basic command line can be extended with options and arguments to

[*path*] `ASReml` [*options*] *basename*[`.as`[`c`]] [*arguments*]

- *options* is a string preceded by a `-` (minus) sign. Its components control several operations (batch, graphic, workspace, . . .) at run time; for example, the command line

  ```
  ASReml -w128 rat.as
  ```

  tells ASReml to run the job `rat.as` with workspace allocation of 128mb,

- *arguments* provide a mechanism (mostly for advanced users) to modify a job at run time; for example, the command line

  ```
  ASReml rat.as alpha beta
  ```

  tells ASReml to process the job in `rat.as` as if it read `alpha` wherever `$1` appears in the file `rat.as`, `beta` wherever `$2` appears and `0` wherever `$3` appears (see below).

11.2.2 Processing a `.pin` file

If the filename argument is a `.pin` file, (see Chapter 13), then ASReml processes it. If the pinfile basename differs from the basename of the output files it is processing, then the basename of the output files must be specified with the `P` option letter. Thus

```
ASReml border.pin
```

will perform the pinfile calculations defined in `border.pin` on the results in files `border.asr` and `border.vvp`.

```
ASReml -Pborderwwt border.pin
```

will perform the pinfile calculations defined in `border.pin` on the results in files `borderwwt.asr` and `borderwwt.vvp`.

11.2.3 Forming a job template from a data file

The facility to generate a template `.as` file was introduced in section 3.4.1. Normally, the name of a `.as` command file is specified on the command line. If a `.as` file does not exist and a file with file extension `.asd`, `.csv`, `.dat`, `.gsh`, `.txt` or `.xls` is specified, ASReml assumes the data file has field labels in the first row and generates a `.as` file template. First, it seeks to convert the `.gsh` (Genstat) or `.xls` (Excel, see page 42) file to `.csv` format. In generating the .as template, ASReml takes the first line of the .csv (or other) file as providing column headings, and generates field definition lines from them. If some labels have ! appended, these are defined as factors, otherwise ASReml attempts to identify factors from the field contents. The template needs further editing before it is ready to run but does have the field names copied across.

11.3 Command line options

Command line options and arguments may be specified on the command line or on the top job control line. This is an optional first line of the .as file which sets command line options and arguments from within the job. If the first line of the .as file contains a qualifier other than !DOPATH, it is interpreted as setting command line options and the *Title* is taken as the next line.

The option string actually used by ASReml is the combination of what is on the command line and what is on the job control line, with options set in both places taking values from the command line. Arguments on the top job control line are ignored if there are arguments on the command line. This section defines the options. Arguments are discussed in detail in a following section.

Command line options are not case sensitive and are combined in a single string preceded by a - (minus) sign, for example -LNW128

The options can be set on the command line or on the first line of the job either as a concatenated string in the same format as for the command line, or as a list of qualifiers. For example, the command line

```
    ASReml -h22r jobname 1 2 3
```

could be replaced with

```
    ASReml jobname
```

if the first line of jobname.as was either

```
!-h22r 1 2 3
```

or

```
!HARDCOPY !EPS !RENAME !ARGS 1 2 3
```

Table 11.1 presents the command line options with brief descriptions. It also gives the name of the equivalent qualifier used on the top job control line. Detailed descriptions follow.

11.3 Command line options

Table 11.1: Command line options

option	qualifier	type	action
Frequently used command line options			
`C`	`!CONTINUE`	job control	continue iterations using previous estimates as initial values
`F`	`!FINAL`	job control	continue for one more iteration using previous estimates as initial values
`L`	`!LOGFILE`	screen output	copy screen output to *basename*`.asl`
`N`	`!NOGRAPHS`	graphics	suppress interactive graphics
`W`*w*	`!WORKSPACE` *w*	workspace	set workspace size to *w* Mbyte
Other command line options			
	`!ARGS` *a*	job control	to set arguments (*a*) in job rather than on command line
`A`	`!ASK`	job control	prompt for options and arguments
`B`*b*	`!BRIEF` *b*	output control	reduce output to `.asr` file
`D`	`!DEBUG`	debug	invoke debug mode
`E`	`!DEBUG 2`	debug	invoke extended debug mode
`G`*g*	`!GRAPHICS` *g*	graphics	set interactive graphics device
`H`*g*	`!HARDCOPY` *g*	graphics	set interactive graphics device, graphics screens not displayed
`I`	`!INTERACTIVE`	graphics	display graphics screen
`O`	`!ONERUN`	job control	override rerunning requested by `!RENAME`
	`!OUTFOLDER`	output control	changes output folder
`P`	NA	post-processing	calculation of functions of variance components
`Q`	`!QUIET`	graphics	suppress screen output
`R`*r*	`!RENAME`	job control	repeat run for each argument renaming output filenames
`S`*s*	NA	workspace	set workspace size
`Y`*v*	`!YVAR` *v*	job control	over-ride *y*-variate specified in the command file with variate number *v*
`Z`	NA	license	reports current license details
`X`	`!XML`	output control	requests that the main output from the `.asr`, `.pvs` and `.sln` files be also written in the `.xml` file.

11.3.1 Prompt for arguments (A)

`A` (`!ASK`) makes it easier to specify command line options in Windows Explorer. One of the options available when right clicking a `.as` file, invokes ASReml with this option. ASReml then prompts for the *options and arguments*, allowing these to be set interactively at run time. With `!ASK` on the top job control line, it is assumed that no other qualifiers are set on the line. For example, a response of

```
     -h22r 1 2 3
```

would be equivalent to

```
ASReml -h22r basename 1 2 3
```

11.3.2 Output control (B, `!OUTFOLDER`, `!XML`)

`B`[*b*] (`!BRIEF` [*b*]) suppresses some of the information written to the `.asr` file. The data summary and regression coefficient estimates are suppressed by the options `B`, `B1` or `B2`. This option should not be used for initial runs of a job before you have confirmed (by checking the data summary) that ASReml has read the data as you intended. Use `B2` to also have the predicted values written to the `.asr` file instead of the `.pvs` file. Use `B-1` to get BLUE estimates reported in `.asr` file.

`!OUTFOLDER` [*path*] allows most of the output files to be written to a folder other than the working folder. This qualifier must be placed on the top command line as it needs to be processed before any output files are opened. Most files produced by ASReml have a filename structure

$<basename><subname>.<extension>$

where $<subname>$ is a command line argument value. If `!OUTFOLDER` is specified without *path*, the output filename pattern becomes

$<basename><subname>/<basename>.<extension>$

If *path* is specified, the output filename pattern becomes

$<path>/<basename><subname>.<extension>$

There are a few files written by ASReml that do not follow this naming pattern, for example, `ainverse.bin` and `asrdata.bin`. These remain unchanged, that is, they are not written to the output folder.

`!XML` requests that the primary tables reported in the `.asr` file and key output from `.pvs` and `.sln` files are written to a `.xml` file in xml format. The output is presented in the order of computation. The first block written is a `.asr` block and includes start and finish times, the data summary, the iteration sequence summary and information criteria, then from the `.pvs` file the tables and associated information, then the summary of estimated variance structure parameters from the `.asr` file, then information from the `.sln` file, and then finally, the Wald F statistics and completion information from the `.asr` file. The process is repeated for each cycle of analysis. The intended use of this file is by programs written to parse ASReml output. For further details, including the status of intended future developments, please contact `support@vsni.co.uk`.

11.3.3 Debug command line options (D, E)

D and E (!DEBUG, !DEBUG 2) invoke debug mode and increase the information written to the screen or .asl file. This information is not useful to most users. On Unix systems, if ASReml is crashing use the system script command to capture the screen output rather than using the L option, as the .asl file is not properly closed after a crash.

11.3.4 Graphics command line options (G, H, I, N, Q)

Graphics are produced by ASReml on some platforms (e.g. PC and Linux) using the Winteracter graphics library.

The I (!INTERACTIVE) option permits the variogram and residual graphics to be displayed. This is the default unless the L option is specified.

The N (!NOGRAPHICS) option prevents any graphics from being displayed. This is the default when the L option is specified.

The G*g* (!GRAPHICS *g*) option sets the file type for hard copy versions of the graphics. Hard copy is formed for all the graphics that are displayed.

H[*g*] (!HARDCOPY *g*) replaces the G option when graphics are to be written to file but not displayed on the screen. The H may be followed by a format code e.g. H22 for .eps.

Q (!QUIET) is used when running under the control of ASReml-W to suppress any POP-UPs/ PAUSES from ASReml.

ASReml writes the graphics to files whose names are built up as ***<basename>[<args>]<type>[<pass>][<section>].<ext>*** where square parentheses indicate elements that might be omitted, <*basename*> is the name portion of the .as file, <*args*> is any argument strings built into the output names by use of the !RENAME qualifier, <*type*> indicates the contents of the figure (as given in the following table), <*pass*> is inserted when the job is repeated (!RENAME or !CYCLE) to ensure filenames are unique across repeats, <*section*> is inserted to distinquish files produced from different sections of data (for example from multisite spatial analysis) and <*ext*> indicates the file graphics format.

<*type*>	file contents
_R_	marginal means of residuals from spatial analysis of a section
_V_	variogram of residuals from spatial analysis for a section
_S_	residuals in field plan for a section
_H_	histogram of residuals for a section
_RvE	residuals plotted against expected values
XYGi	figure produced by !X, !Y and !G qualifiers
PV_i	Predicted values plotted for PREDICT directive *i*

The graphics file format is specified by following the G or H option by a number *g*, or specifying the appropriate qualifier on the top job control line, as follows:

g	qualifier	description	<*ext*>
1	`!HPGL`	HP-GL	`pgl`
2	`!PS`	Postscript (default)	`ps`
6	`!BMP`	BMP	`bmp`
10	`!WPM`	Windows Print Manager	
11	`!WMF`	Windows Meta File	`wmf`
12	`!HPGL 2`	HP-GL2	`hgl`
21	`!PNG`	PNG	`png`
22	`!EPS`	EncapsulatedPostScript	`eps`

11.3.5 Job control command line options (C, F, O, R)

`C` (`!CONTINUE`) indicates that the job is to continue iterating from the values in the `.rsv` file. This is equivalent to setting `!CONTINUE` on the datafile line, see Table 5.4, page 66 for details.

`F` (`!FINAL`) indicates that the job is to continue for one more iteration from the values in the `.rsv` file. This is useful when using `predict`, see Chapter 10.

`O` (`!ONERUN`) is used with the `R` option to make ASReml perform a single analysis when the `R` option would otherwise attempt multiple analyses. The `R` option then builds some arguments into the output file name while other arguments are not. For example

```
ASReml -nor2 mabphen 2 TWT out(621) out(929)
```

results in one run with output files `mabphen2_TWT.*`.

`R[`r`]` (`!RENAME [`r`]`) is used in conjunction with at least r argument(s) and does two things: it modifies the output filename to include the first r arguments so the output is identified by these arguments, and, if there are more than r arguments, the job is rerun moving the extra arguments up to position r (unless `!ONERUN` (`O`) is also set). If r is not specified, it is taken as 1.

For example

```
    ASReml -r2 job wwt gfw fd fat
```

is equivalent to running three jobs:

```
     ASReml -r2 job wwt gfw → jobwwt_gfw.asr
     ASReml -r2 job wwt fd → jobwwt_fd.asr
     ASReml -r2 job wwt fat → jobwwt_fat.asr
```

`Y`y (`!YVAR` y) overrides the value of *response*, the variate to be analysed (see Section 6.2) with the value y, where y is the *number* of the data field containing the trait to be analysed. This facilitates analysis of several traits under the same model. The value of y is appended to the *basename* so that output files are not overwritten when the next trait is analysed.

11.3.6 Workspace command line options (S, W)

The workspace requirements depend on problem size and may be quite large. On 32bit computers the maximum is 2000Mbyte under Linux, 1600 Mbyte under Windows. On 64bit systems, the maximum is 32 Gbyte but may be less depending on the machine configuration. The default allocation is 32Mbyte (4 million double precision words). An increased workspace allocation may be requested on the command line with the W*m* option.

W*m* (!WORKSPACE *m*) sets the initial size of the workspace in Mbytes. For example W1600 requests 1600 Mbytes of workspace, the maximum typically available under Windows. W2000 is the maximum available on 32bit Unix(Linux) systems. On 64bit systems, the argument, if less than 33, is taken as Gbyte.

If your system cannot provide the requested workspace, the request will be diminished until it can be satisfied. On multi-user systems, do not unnecessarily request the maximum or other users may complain.

Having started with an initial allocation, if ASReml realises more space is required as it is running, it will attempt to restart the job with increased workspace. If the system has already allocated all available memory the job will stop.

11.3.7 Examples

ASReml code	action
`asreml -LW64 rat.as`	increase workspace to 64 Mbyte, send screen output to `rat.asl` and suppress interactive graphics
`asreml -IL rat.as`	send screen output to `rat.asl` but display interactive graphics
`asreml -N rat.as`	allow screen output but suppress interactive graphics
`asreml -ILW512 rat.as`	increase workspace to 512 Mbyte , send screen output to `rat.asl` but display interactive graphics
`asreml -rw1 coop wwt ywt`	runs `coop.as` twice using 1Gbyte workspace and writing results to `coopwwt.as` and `coopywt.as` and substituting `wwt` and `ywt` for `$1` in the two runs.

11.4 Advanced processing arguments

11.4.1 Standard use of arguments

Command line arguments are intended to facilitate the running of a sequence of jobs that require small changes to the command file between runs. The output file name is modified by the use of this feature if the -R option is specified. This use is demonstrated in the Coopworth example of Section 16.10.

Command line arguments are strings listed on the command line after *basename*, the command file name, or specified on the top job control line after the !ARGS qualifier. These strings are inserted into the command file at run time. When the input routine finds a $\$n$ in the command file it substitutes the *n*th argument (string). *n* may take the values 1. . .9 to indicate up to 9 strings after the command file name. If the argument has 1 character, a trailing blank is attached to the character and inserted into the command file. If no argument exists, a zero is inserted. For example,

```
asreml rat.as alpha beta
```

tells ASReml to process the job in rat.as as if it read alpha wherever $1 appears in the command file, beta wherever $2 appears and 0 wherever $3 appears.

Table 11.2: The use of arguments in ASReml

in command file	on command line	becomes in ASReml run
abc$1def	no argument	abc0 def
abc$1def	with argument X	abcX def
abc$1def	with argument XY	abcXYdef
abc$1def	with argument XYZ	abcXYZdef
abc$1 def	with argument XX	abcXX def
abc$1 def	with argument XXX	abcXXX def
abc$1 def (multiple spaces)	with argument XXX	abcXXX def

11.4.2 Prompting for input

Another way to gain some interactive control of a job in the PC environment is to insert !?{*text*} in the .as file where you want to specify the rest of the line at run time. ASReml prompts with *text* and waits for a response which is used to compete the line. The !? qualifier may be used anywhere in the job and the line is modified from that point.

Warning Unfortunately the prompt may not appear on the top screen under some windows operating systems in which case it may not be obvious that ASReml is waiting for a keyboard response.

11.4.3 Paths and Loops

ASReml was designed to analyse just one model per run. However, the analysis of a data set typically requires many runs, fitting different models to different traits. It is often convenient to have all these runs coded into a single `.as` file and control the details from the command line (or top job control line) using arguments. The highlevel qualifiers `!CYCLE` and `DOPATH` enable multiple analyses to be defined and run in one execution of ASReml.

Table 11.3: High level qualifiers

qualifier	action

`!ASSIGN` *list*	New R4 An `!ASSIGN` *string* qualifier has been added to extend coding options. It is a high level qualifier command which may appear anywhere in the job. Each occurrence of `!ASSIGN` must start on its own input line. The syntax is
	`!ASSIGN` *name string*
	or
	`!ASSIGN` *name* !< *string* !>
	and the defined *string* is substituted into the job where `$`*name* appears. *string* is the rest of the line and may include blanks. If !< !> encloses *string*, *string* may extend over several lines, which are concatenated.
	For example `!ASSIGN TVS xfa1(Treat)`
	`...`
	`...  $TVS.geno ...`
	is interpreted as
	`...  xfa1(Treat).geno ...`
	Restrictions:
	• a maximum of 50 assign strings may be defined.
	• the combined length of all strings is 5000 characters.
	• *name* may have up to 8 characters but should not begin with a number (see command line arguments).
	• dollar substitution occurs before most other high level actions. ASSIGN strings and commandline arguments may substitute into a `!CYCLE` line.
	• `I`, `J`, `K` and `L` are reserved as names referring to items in the `!CYCLE` list and should therefore not be used as names of an ASSIGN string.

High level qualifiers

qualifier — action

`!CYCLE [!SAMEDATA]` *list*

is a mechanism whereby ASReml can loop through a series of jobs. The `!CYCLE` has a qualifier `!SAMEDATA` that tells ASReml to use the same data for all cycles, ie. the data file is only read on the first cycle, and is kept in memory for later cycles. The `!CYCLE` qualifier must appear on its own line. *list* is a series of values which are substituted into the job wherever the `$I` string appears. The list may spread over several lines if each incomplete line ends with a COMMA. A series of sequential integer values can be given in the form $i:j$ (no embedded spaces). The output from the set of runs is concatenated into a single set of files, but the output written to the `.asr` file is slightly abbreviated after the first cycle, by suppressing the data summary and fixed effect solutions that might otherwise appear (see `!BRIEF`; the `!BRIEF` qualifier is set after the first cycle).

For example

```
!CYCLE 0.4 0.5 0.6
20 0 mat2 1.9 $I !GPF
```

would result in three runs and the results would be appended to a single file. Putting `!SAMEDATA` on the (leading) `!CYCLE` line makes ASReml read the data (and `.grr` file) file in the first `CYCLE` and hold it in memory for use in subsequent cycles. This is advantageous when the data/`.grr` file is large and there are many cycles to execute where the model changes, but the data/`.grr` file doesn't.

The `!CYCLE` mechanism acts as an inner loop when used with `!RENAME !ARG`. As an example, the `!RENAME !ARG` arguments might list a set of traits, and the `!CYCLE` arguments sequentially test a set of markers.

A cycle string may consist of up to 4 substrings, separated by a semicolon and referenced as `$I $J $K` and `$L` respectively. For example

```
 !CYCLE Y1;X1 Y2;X2
$I ~ mu $J
```

When cycling is active, an extra line is written to the `.asr` file containing some details of the cycle in a form which can be extracted to form an analysis summary by searching for `LogL:`. A heading for this extra line is written in the first cycle. For example

```
 LogL: LogL Residual NEDF NIT Cycle Text
LogL: -208.97 0.703148 587 6 1466 "LogL Converged"
```

The `LogL:` line with the highest LogL value is repeated at the end of the `.asr` file.

`!DOPATH` *n*
`!DOPART` *n*

`!DOPATH` with `!PATH/!PART` statements allows several analyses to be coded in one job file and run selectively without having to edit the `.as` file between runs. Both spellings can be used interchangably. Which particular lines in the `.as` file are honoured is controlled by the argument *n* of the `!DOPATH` qualifier in conjunction with `!PATH` (or `!PART`) statements.

High level qualifiers

qualifier — action

The argument (n) is often given as `$1` indicating that the actual path to use is specified as the first argument on the command line (see Section 11.4). See Sections 16.7 and 16.10 for examples. The default value of n is 1.

`!DOPATH` n can be located anywhere in the job but if placed on the top job control line, it cannot have the form `!DOPATH $1` unless the arguments are on the command line as the `!DOPATH` qualifier will be parsed before any job arguments on the same line are parsed.

`!FOR` *`forlist`* `!DO` *`command`*

New R4 The `!FOR ... !DO ...` command is intended to simplify coding when a series of similar lines are required in the command file which differ in a single argument. The list of arguments is placed after `!FOR` and the command is written after `!DO` with `$S` indicating where the argument is to be inserted. *list* may be an assign string since they are processed before the `!FOR` statement is expanded. Furthermore, if *list* is entirely integer numbers, i:j notation can be used.

For example

```
!ASSIGN Markern 35 75 125
!ASSIGN Markers M35 M75 M125
!FOR $Markern !DO !MBF mbf(Geno,1) markers.csv !key 1 !RFIELD$S !RENAME M$S
...   ...  !r $Markers
```

is expanded to

```
!MBF mbf(Geno,1) markers.csv !key 1 !RFIELD 35 !RENAME M35
!MBF mbf(Geno,1) markers.csv !key 1 !RFIELD 75 !RENAME M75
!MBF mbf(Geno,1) markers.csv !key 1 !RFIELD 125 !RENAME M125
...    ...  !r M35 M75 M125
```

The aim here is to generate the 3 !MBF statements required to extract markers 35, 75 and 125 from the marker file *markers.csv*. The names of model terms must begin with a letter, hence the marker names are the letter `M` followed by the position number. Alternatively `!RFIELD`*lettersinteger* is interpreted as `!RFIELD` *integer* so the `!FOR` statement can be written even more concisely as

```
!FOR $Markers !DO !MBF mbf(Geno,1) markers.csv !key 1 !RFIELD$S !RENAME $S
```

without the need to assign `Markern`. Now, to add another marker to the model, one can just add the marker integer to the `ASSIGN` statement.

Restriction: *forlist* and *command* are both limited to 200 characters.

High level qualifiers

qualifier	action

`!IF` *string1* `==` *string2 text*	New R4 One form of the `IF` statement is
	`!IF string1 == string2 !ASSIGN M1 brt DamAge` which makes the `!ASSIGN` statement active if `string1` is the same as `string2`. Note that there need to be spaces before and after `==` to avoid confusion with the strings. This has been used when performing a large number of bivariate analyses with trait specific fixed effects being fitted. So
	`...`
	`!IF $1 == wwt !ASSIGN M1 brt DamAge`
	`!IF $1 == ywt !ASSIGN M1 brt`
	`!IF $1 == fwt !ASSIGN M1 DamAge`
	`!IF $2 == wwt !ASSIGN M2 brt DamAge`
	`!IF $2 == ywt !ASSIGN M2 brt`
	`!IF $2 == fwt !ASSIGN M2 DamAge`
	
	`...`
	`$1 $2 ~ Trait at(Trait,1).($M1) at(Trait,2).($M2)`

`!PATH` *pathlist*	The `!PATH` (or `!PART`) control statement may list multiple path numbers so that the following lines are honoured if any one of the listed path numbers is active. The `!PATH` qualifier must appear at the beginning of its own line after the `!DOPATH` qualifier. A sequence of path numbers can be written using $a : b$ notation. For example
	`mydata.asd !DOPATH 4`
	`!PATH 2 4 6:10`
	One situation where this might be useful is where it is necessary to run simpler models to get reasonable starting values for more complex variance models. The more complex models are specified in later parts and the `!CONTINUE` command is used to pick up the previous estimates.

Example

The following code will run through 1000 models fitting 1000 different marker variables to some data. For processing efficiently the 1000 marker variables are held in 1000 separate files in subfolder `MLIB` and indexed by `Genotype`.

```
Marker screen
 Genotype *
 yield
PhenData.txt
!CYCLE 1:1000
!MBF  mbf(Genotype)  MLIB\Marker$I.csv !RENAME Marker$I
 yld ~ mu  !r Marker$I
```

Having completed the run, the Unix command sequence

```
  grep LogL: screen.asr | sort > screen.srt
```

sorts a summary of the results to identify the best fit. The best fit can then be added to the model and the process repeated. Assuming Marker35 was best, the revised job could be

```
Marker screen
 Genotype *
 yield
PhenData.txt
!CYCLE 1:1000
!MBF  mbf(Genotype)  MLIB\Marker$I.csv !RENAME Marker$I
!MBF  mbf(Genotype)  MLIB\Marker35.csv !RENAME MKR035
 yld ~ mu  !r MKR035 Marker$I
```

We have given Marker35 a new name because it is still also generated by the !CYCLE unless it is modified to read
!CYCLE 1:34 36:1000 .

After several cycles, we might have

```
Marker screen
 Genotype *
 yield
PhenData.txt
!ASSIGN MSET R21 R35 R376 R645 R879
!CYCLE 1:1000
!MBF  mbf(Genotype)  MLIB\Marker$I.csv !RENAME Marker$I
!FOR $MSET !DO !MBF  mbf(Genotype)  MLIB\Marke$S.csv !RENAME $S
 yld ~ mu  !r $MSET Marker$I
```

11.4.4 Order of Substitution

The substitution order is ASSIGN, FOR, CYCLE, TP, command line arguments and finally the interactive prompt.

11.5 Performance issues

11.5.1 Multiple processors

ASReml has not been configured for parallel processing. Performance is downgraded if it tries to use two processors simultaneously as it wastes time swapping between processors.

11.5.2 Slow processes

The processing time is related to the size of the model, the complexity of the variance model (in particular the number of parameters), the sparsity of the mixed model equations, the amount of data being processed.

Typically, the first iteration take longer than other iterations. The extra work in the first iteration is to determine an optimum equation order for processing the model (see `!EQORDER`).

The extra processes in the last iteration are optional. They include

- calculation of predicted values (see `PREDICT` statement),
- calculation of denominator degrees of freedom (see `!DDF`),
- calculation of outlier statistics (see `!OUTLIER`).

If a job is being run a large number of times, significant gains in processing time can sometimes be made by reorganising the data (so reading of irrelevant data is avoided), using binary data files, use of `!CONTINUE` to reduce the number of iterations, and avoiding unnecessary output (see `!SLNFORM`, `!YHTFORM` and `!NOGRAPHICS`).

11.5.3 Timing processes

The elapsed time for the whole job can be calculated approximately by comparing the start time with the finish time. Timings of particular processes can be obtained by using the `!DEBUG !LOGFILE` qualifiers on the first line of the job. This requests the `.asl` file be created and hold some intermediate results, especially from data setup and the first iteration. Included in that information is timing information on each phase of the job.

12 Command file: Merging data files

12.1 Introduction

The MERGE directive, described in this chapter, is designed to combine information from two files into a third file with a range of qualifiers to accomodate various scenarios. It was developed with assistance from Chandrapal Kailasanathan to replace the !MERGE qualifier (see page 64) which had very limited functionality.

The MERGE **directive** is placed BEFORE the data filename lines. It is an independent part of the ASReml job in the sense that none of the files are necessarily involved in the subsequent analyses performed by the job, and there may be multiple MERGE directives. Indeed, the job may just consist of a title line and MERGE directives. The !MERGE **qualifier**, on the other hand, combines information from two files into the internal data set which ASReml uses for analysis and does not save it to file. It has very limited in functionality.

The files to be merged must conform to the following basic structure:

- the data fields must be TAB, COMMA or SPACE separated,
- there will be one heading line that names the columns in the file,
- the names may not have embedded spaces,
- the number of fields is determined from the number of names,
- missing values are implied by adjacent commas in comma delimited files. Otherwise, they are indicated by NA, * or . as in normal ASReml files.
- the merged file will be TAB separated if a .txt file, COMMA separated if a .csv file and SPACE separated otherwise.

12.2 Merge Syntax

The basic merge command is

12.2 Merge Syntax

`MERGE` *file1* `!WITH` *file2* `!TO` *newfile*.

Typically files to be merged will have common *key* fields. In the basic merge, (`!KEY` not specified) any fields having the same names are taken as the *key* fields and if the files have no fields in common, they are assumed to match on row number. Fields are referenced by name (case sensitive).

The full command is:

```
MERGE file1 [ !KEY keyfields ] [ !KEEP ] [ !SKIP fields ]
      !WITH file2 [ !KEY keyfields ] [ !KEEP ] [ !NODUP ] [ !SKIP fields]
      !TO newfile [!CHECK ] [ !SORT ].
```

Warning: Fields in the merged file will be arranged with key fields followed by other fields from the primary file and then fields from the secondary file.

Table 12.1: List of MERGE qualifiers

qualifier	action
`!CHECK`	requests ASReml confirm that fields having a common name have the same contents. Discrepancies are reported to the `.asr` file. If there are fields with common names which are not *key* fields, and `!CHECK` is omitted, the fields will be assumed different and both versions will be copied.
`!KEY` *`keyfields`*	names the fields which are to be used for matching records in the files. If the fields have the same name in both file headers, they need only be named in association with the primary input file. If the key fields are the only fields with common names, the `!KEY` qualifier may be omitted altogether. If key fields are not nominated and there are no common field names, the files are interleaved.
`!KEEP`	instructs ASReml to include in the merged file records from the input file which are not matched in the other input file. Missing values are inserted as the values from the other file. Otherwise, unmatched records are discarded. `!KEEP` may be specified with either or both input files.
`!NODUP` *`fields`*	Typically when a match occurs, the field contents from the second file are combined with the field contents of the first file to produce the merged file. The `!NODUP` qualifier, which may only be associated with the second file, causes the field contents for the nominated fields from the second file only be inserted once into the merged file. For example, assume we want to merge two files containing data from sheep. The first file has several records per animal containing fleece data from various years. The second file has one record per animal containing birth and weaning weights. Merging with `!NODUP bwt wwt` will copy these traits only once into the merged file.
`!SKIP` *`fields`*	is used to exclude fields from the merged file. It may be specified with either or both input files.
`!SORT`	instructs ASReml to produce the merged file sorted on the key fields. Otherwise the records are return in the order they appear in the primary file.

The merging algorithm is briefly as follows: The secondary file is read in, *skip* fields being omitted, and the records are sorted on the *key* fields. If sorted output is required, the primary file is also read in and sorted. The primary file (or its sorted form) is then processed line by line and the merged file is produced. Matching of key fields is on a string basis, not a value basis. If there are no key fields, the files are merged by interleaving.

If there are multiple records with the same key, these are severally matched. That is if 3 lines of file 1 match 4 lines of file 2, the merged file will contain all 12 combinations.

12.3 Examples

Key fields have different names

`!MERGE` *file1* `!KEY` *key1a key1b* `!WITH` *file2* `!KEY` *key2a key2b* `!TO` *newfile*

Key fields have common name and other fields are also duplicated

`!MERGE` *file1* `!KEY` *keya keyb* `!WITH` *file2* `!TO` *newfile* `!CHECK`

`!MERGE` *file1* `!Key` *key* `!KEEP !WITH` *file2* `!to` *newfile*

will discard records from *file2* that do not match records in *file1* but all records in `file1` are retained.

Omitting fields from the merged file

`!MERGE` *file1* `!KEY` *key* `!skip` *s1a s1b* `!WITH` *file2* `!SKIP` *s2a s2b* `!TO` *newfile*

Single insertion merging

`!MERGE` *adult.txt* `!KEY` *ewe* `!KEEP !WITH` *birth.txt* `!KEEP !TO` *newfile* `!NODUP bwt`.

13 Functions of variance components

13.1 Introduction

ASReml includes a procedure to calculate certain functions of variance components either as a final stage of an analysis or as a post-analysis procedure. These functions enable the calculation of heritabilities and correlations from simple variance components and when US, CORUH and XFA structures are used in the model fitting. A simple example is shown in the code box. The instructions to perform the required operations are listed after the VPREDICT !DEFINE line and terminated by a blank line. ASReml holds the instructions in a .pin until the end of the job when it retrieves the relevant information from the .asr and .vvp files and performs the specified operations. The results are reported in the .pvc file.

```
y ~ mu !r idv(Sire)
residual idv(units)
VPREDICT !DEFINE
F phenvar idv(Sire) + idv(units)
F genvar idv(Sire) * 4
R herit genvar phenvar
```

In Section 13.2 the syntax for these instructions are discussed. Direct use of the .pin file, as was required in ASReml 2, is discussed in Section 13.3.

13.2 Syntax

Instructions to calculate functions are headed by a line

```
VPREDICT !DEFINE
```

This line and the following instructions can occur anywhere in the .as file but the logical place is at the end of the file. The instructions are processed after the job (part/cycle) has been completed. ASReml recognises a blank line (or end of file) as termination of the functional instructions.

Functions of the variance components are specified by lines of the form

letter label coefficients

- *letter* (either F, H, R, S, V or X) must occur in column 1
  - F forms linear combinations of variance components,

  - H is for forming heritabilities, the ratio of two components,
  - R is for forming the correlation from a covariance component,
  - S is a square root function,
  - V is for converting components related to a CORUH or an XFA structure into components related to a US structure,
  - X is a multiply function,

- *label* names the result,
- *coefficients* is the list of arguments/coefficients for the linear function.

When ASReml reads back the variance parameters from the `.asr` file, each covariance component, or variance function, is assigned a name. The full name is usually the covariance function, or its specified contracted form, prepended by the consolidated model term, or its specified contracted form, and the symbol ;. Exceptions to this rule are single components `F`, `id[v](F)` and `nrm[v](F)` terms which are reduced to the corresponding single term `F`, `id[v](F)` and `nrm[v](F)`. So, for example, with the random model and residual specification model terms

```
!r idv(A) ar1v(B) nrm(C).us(Trait) D
residual id(units).us(Trait)
```

The covariance functions with parameters

`idv(A),ar1v(B), us(Trait)` in `nrm(C).us(Trait)`

and

`us(Trait)` in `id(units).us(Trait)` are named

`idv(A), ar1v(B);ar1v(B), nrm(C).us(Trait);us(Trait), id(units).us(Trait);us(Trait)`. If the resulting name is not ambiguous the name can be contracted by reducing the consolidated model term to a unique substring or leaving out the consolidated model term completely. For example, in the example the covariance functions can be represented by `idv(A)`, `ar1v(B)`, `C;us(Trait)` and `units;us(Trait)`, respectively. Individual parameters within a covariance component can be specified by number , or sequence of numbers (`n:m`) by appending these in square braces, for example, `C;us(Trait)[3]` or `units;us(Trait)[4:6]`. If the residual directive is not used, the default R structure parameters are effectively named `Residual`. The orphan term `D` with no explicit variance function is treated as `idv(D)` structure with name `D`. If the user is in doubt of the name or number of a parameter then running the program with `VPREDICT !DEFINE` and a blank line will construct a `.pvc` file with the names and numbers of parameters identified.

The original implementation was based entirely on the numbers but it will generally be better to use the names, since the order model terms are reported cannot always be predicted.

Critical change For generalised linear models in ASReml Release 4, the `.pvc` file reports and numbers, for completeness, a residual or dispersion parameter both when the parameter is estimated or when it is fixed. By contrast, ASReml 3 does not report nor number if the parameter is fixed by default at 1. Hence the parameters might be numbered differently in ASReml 4 and ASReml 3.

13.2.1 Functions of components

```
y ~ mu !r idv(Sire)
residual idv(units)
VPREDICT !DEFINE
F phenvar idv(Sire) + idv(units)
F genvar idv(Sire) * 4
R herit genvar phenvar
```

First ASReml extracts the variance components from the `.asr` file and their variance matrix from the `.vvp` file. The `F`, `S`, `V` and `X` functions create new components which are appended to the list. For example, the `F` function appends component $k + \boldsymbol{c}'\boldsymbol{v}$ and forms $\text{cov}\,(\boldsymbol{c}'\boldsymbol{v}, \boldsymbol{v})$ and $\text{var}\,(\boldsymbol{c}'\boldsymbol{v})$ where $\boldsymbol{v}$ is the vector of existing variance components, $\boldsymbol{c}$ is the vector of coefficients for the linear function and k is an optional offset which is usually omitted but would be 1 to represent the residual variance in a probit analysis and 3.289 to represent the residual variance in a logit analysis. The general form of the directive is

`F` *label* $a \; + \; b \; * \; c_b \; + \; c \; + \; d \; + m * k$

where a, b, c and d are the numbers or names of existing components v_a, v_b, v_c and v_d and c_b is a multiplier for v_b. m is a number greater than the current length of $\boldsymbol{v}$ to flag the special case of adding the offset k. When using the component numbers, the form *a*:*b* can be used to reference blocks of components as in

`F` *label* *a*:*b* $^*\, k + c{:}d$

The instructions in the ASReml code box corresponds to a simple sire model so that variance component 1 is the Sire variance and variance component 2 is the residual variance, then

```
F phenvar  1 + 2
```

or

```
F phenvar  idv(Sire) + idv(units)
```

creates a third component called `phenvar` which is the sum of the variance components, that is, the phenotypic variance,

```
F genvar  1 * 4
```

or

```
F genvar  idv(Sire) * 4
```

creates a fourth component called `genvar` which is the sire variance component multiplied by 4, that is, the genotypic variance.

Ratios, or in particular cases heritabilities, are requested by function lines beginning with an `H`. The specific form of the directive is

`H` *label n d*

This calculates σ^2_n/σ^2_d and $se[\sigma^2_n/\sigma^2_d]$ where n and d are the names of the components or integers pointing to components v_n and v_d that are to be used as the numerator and denominator respectively in the heritability calculation.

```
y ~ mu !r idv(Sire)
residual idv(units)
VPREDICT !DEFINE
F phenvar idv(Sire) +idv(units)
F genvar idv(Sire) * 4
R herit genvar phenvar
```

Note that covariances between ratios and other components are not generated so the ratios are not numbered and cannot be used to derive other functions. To avoid numbering confusion it is better to include `H` functions at the end of the `VPREDICT` block.

In the example

`H herit  4 3` or `H herit genvar phenvar`

calculates the heritability by calculating component 4 (from second line) / component 3 (from first line), that is, *genetic variance / phenotypic variance.*

`S` *label i:j* when *i:j* are assumed positive variance parameters, inserts components which are the `SQRT` of components *i:j*.

`X` *label i*k* inserts a component being the product of components i and k.

`X` *label i:j*k* inserts $j - i + 1$ components being the products of components $i : j$ and k.

`X` *label i:j*k:l* inserts a set of $j-i+1$ components being the pairwise products of components $i : j$ and $k : l$.

The `S` and `X` functions are new in ASReml Release 4. The multiply option (`X`) allows a correlation in a CORUV structure to be converted to a covariance. The `SQRT` option allows conversion of CORGH to US, provided the dimension is moderate (say < 10).

The variances and covariances are calculated using a Taylor series expansion. Then for parameters v_a and v_b derived from the set of parameters $\boldsymbol{v}$ with variance matrix $\boldsymbol{V}$, if $v_a = f_a(\boldsymbol{v})$ and $v_b = f_b(\boldsymbol{v})$ then if $\delta\boldsymbol{v}_a = \frac{\delta f_a(\boldsymbol{v})}{\delta v}$ and if $\delta\boldsymbol{v}_b = \frac{\delta f_b(\boldsymbol{v})}{\delta v}$ then $cov(v_a, v_b) = \delta\boldsymbol{v}'_a\boldsymbol{V}\delta\boldsymbol{v}_b$.

13.2.2 Convert CORUH and XFA to US

`V` *label i:j* where $i : j$ spans a CORUH variance structure, inserts the US matrix based on the CORUH parameters.

`V` *label i:j* where $i : j$ spans an XFA variance structure, inserts the US matrix based on the XFA parameters.

13.2.3 Correlation

```
y1 y2 ~ Trait !r id(sire).us(Trait)
residual id(units).us(Trait)
VPREDICT !DEFINE
F phenvar 4:6 + 1:3
#id(sire).us(Trait);us(Trait)
#+units;us(Trait)
R phencorr 7:9 #phenvar
R gencorr 4:6 #sire;us(Trait)
```

Correlations are requested by lines beginning with an R. The specific form of the directive is

`R` *label a ab b*

This calculates the correlation $r = \sigma_{ab}/\sqrt{\sigma_a^2\sigma_b^2}$ and the associated standard error. a, b and ab are integers indicating the position of the components to be used. Alternatively,

`R` *label a:n*

calculates the correlation $r = \sigma_{ab}/\sqrt{\sigma_a^2\sigma_b^2}$ for all correlations in the lower triangular row-wise matrix represented by components a to n and the associated standard errors.

Note that covariances between ratios and other components are not generated so the correlations are not numbered and cannot be used to derive other functions. To avoid numbering confusion it is better to include `R` functions at the end of the `VPREDICT` block.

In the example

`R phencorr  7 8 9` or `R phencorr  phenvar`

calculates the phenotypic covariance by calculating component 8 / $\sqrt{\text{component 7} \times \text{component 9}}$ where components 7, 8 and 9 are created with the first line of the `.pin` file, and

`R gencorr  4:6` or `R gencorr  sire;us(Trait)`

calculates the genotypic covariance by calculating component 5 / $\sqrt{\text{component 4} \times \text{component 6}}$ where components 4, 5 and 6 are variance components from the analysis.

13.2.4 A more detailed example

The following example for a **bivariate sire model** is a little more complicated. The job file `bsiremod.as` contains

```
...
coop.fmt

ywt fat ~ Trait Trait.(age c(brr) sex sex.age) !r us(Trait).id(sire);us(Trait)  !f Tr.grp
residual id(units).us(Trait)

VPREDICT !DEFINE
F phenvar id(units).us(Trait);us(Trait) + us(Trait).sire;us(Trait) # 1:3 + 4:6
F addvar sire;us(Trait) * 4                                         # 4:6 * 4
H heritA addvar[1] phenvar[1]                                       # 10 7
H heritB addvar[3] phenvar[3]                                       # 12 9
R phencorr phenvar                                                  # 7 8 9
R gencorr addvar                                                    # 4:6
```

The relevant lines of the `.asr` file are

```
Model_Term                                  Sigma         Sigma   Sigma/SE   % C
id(units).us(Trait)                   8140 effects
Trait                       US_V  1  1    23.2055       23.2055      44.44   0 P
Trait                       US_C  2  1    2.50402       2.50402      18.56   0 P
Trait                       US_V  2  2    1.66292       1.66292      32.82   0 P
us(Trait).id(sire)                     184 effects
Trait                       US_V  1  1    1.45821       1.45821       3.66   0 P
Trait                       US_C  2  1   0.130280      0.130280       1.92   0 P
Trait                       US_V  2  2   0.344381E-01  0.344381E-01   2.03   0 P
```

Numbering the parameters reported in `bsiremod.asr` (and `bsiremod.vvp`)

1 error variance for `ywt`
2 error covariance for `ywt` and `fat`
3 error variance for `fat`
4 sire variance component for `ywt`
5 sire covariance for `ywt` and `fat`
6 sire variance for `fat`

then

`F phenvar id(units).us(Trait);us(Trait) + us(Trait).id(sire);us(Trait)` or
`F phenvar units;us(Trait) + sire;us(Trait)` or `F phenvar 1:3 + 4:6`

creates new components **7** = **1**+**4**, **8** = **2**+**5** and **9** = **3**+**6**,

`F addvar sire;us(Trait) * 4` or `F addvar 4:6 * 4`

creates new components **10** = **4** × 4, **11** = **5** × 4 and **12** = **6** × 4,

`H heritA addvar[1] phenvar[1]` or `H heritA 10 7`

forms **10** / **7** to give the heritability for `ywt`,

`H heritB addvar[3] phenvar[3]` or `H heritB 12 9`

forms **12** / **9** to give the heritability for `fat`,

`R phencorr phenvar` or `R phencorr 7 8 9`

forms $\mathbf{8} / \sqrt{\mathbf{7} \times \mathbf{9}}$, that is, the phenotypic correlation between `ywt` and `fat`,

`R gencorr addvar` or `R gencorr 4:6`

forms $\mathbf{5} / \sqrt{\mathbf{4} \times \mathbf{6}}$, that is, the genetic correlation between `ywt` and `fat`.

The resulting `.pvc` file contains:

```
 id(units).us(Trait)            8140 effects
   1 id(units).us(Trait);us(Trait)          V  1  1      23.2055        0.522176
   2 id(units).us(Trait);us(Trait)          C  2  1      2.50402        0.134915
   3 id(units).us(Trait);us(Trait)          V  2  2      1.66292        0.506679E-01
 us(Trait).id(sire)              184 effects
   4 us(Trait).id(sire);us(Trait)           V  1  1      1.45821        0.398418
   5 us(Trait).id(sire);us(Trait)           C  2  1     0.130280        0.678542E-01
   6 us(Trait).id(sire);us(Trait)           V  2  2     0.344381E-01    0.169646E-01
   7 phenvar  1                 24.664       0.64250
   8 phenvar  2                 2.6343       0.14763
   9 phenvar  3                 1.6974       0.52365E-01
  10 addvar  4                  5.8328        1.5926
  11 addvar  5                 0.52112       0.27168
  12 addvar  6                 0.13775       0.67791E-01
     heritA       = addvar    10/phenvar    7=          0.2365    0.0612
     heritB       = addvar    12/phenvar    9=          0.0812    0.0394
     phenco  2  1 = phenv  8/SQR[phenv  7*phenv  9]=    0.4071    0.0183
     gencor  2  1 = addva 11/SQR[addva 10*addva 12]=    0.5814    0.2039
 Notice: The parameter estimates are followed by
          their approximate standard errors.
```

The first 8 lines are based on the `.asr` file.

13.3 VPREDICT: PIN file processing

There are four forms of the `VPREDICT` directive.

- If the `.pin` file exists and has the same name as the jobname (including any suffix appended by using `!RENAME`), just specify the `VPREDICT` directive.

- If the `.pin` file exists but has a different name to the jobname, specify the `VPREDICT` directive with the `.pin` file name as its argument.

- If the `.pin` file does not exist or must be reformed, a name argument for the file is optional but the `!DEFINE` qualifier should be set. Then the lines of the `.pin` file should follow on the next lines, terminated by a blank line.

An alternative to using `VPREDICT` is process the contents of the `.pin` file by running ASReml with the -P command line option specifying the `.pin` file as the input file.

Note that in this case the code must be self contained and any substitution variable used needs defining in the `.pin` file. For example, if we wish to use `$sub` to indicate `fullname`, then the assignment of `fullname` to `sub` using

```
!ASSIGN sub fullname
```

needs to be in the `.pin` file.

14 Description of output files

14.1 Introduction

With each ASReml run a number of output files are produced. ASReml generates the output files by appending various filename extensions to *basename*. A brief description of the filename extensions is presented in Table 14.1.

Table 14.1: Summary of ASReml output files

file	description
Key output files	
`.asr`	contains a summary of the data and analysis results.
`.msv`	contains final variance parameter values in a form that is easy to edit for resetting the initial values if `!MSV` or `!CONTINUE 3` is used, see Table 5.4.
`.pvc`	contains the report produced with the `P` option.
`.pvs`	contains predictions formed by the `predict` directive.
`.res`	contains information from using the `pol()`, `spl()` and `fac()` functions, the iteration sequence for the variance components and some statistics derived from the residuals.
`.rsv`	contains the final parameter values for reading back if the `!CONTINUE` qualifier is invoked, see Table 5.4.
`.sln`	contains the estimates of the fixed and random effects and their corresponding standard errors.
`.tab`	contains tables formed by the `tabulate` directive.
`.tsv`	contains variance parameter values in a form that is easy to edit for resetting the initial values if `!TSV` or `!CONTINUE 2` is used, see Table 5.4.
`.yht`	contains the predicted values, residuals and diagonal elements of the hat matrix for each data point.
Other output files	
`.asl`	contains a progress log and error messages if the `L` command line option is specified.
`.aov`	contains details of the ANOVA calculations.
`.apj`	is an ASReml project file created by ASReml-W .

14.1 Introduction

Table 14.1: Summary of ASReml output files

file	description
.ask	holds the !RENAME !ARG argument from the most recent run so that ASReml can retrieve restart values from the most recent run when !CONTINUE is specified but there is no particular .rsv file for the current !ARG argument.
.asp	contains transformed data, see !PRINT in Table 5.2.
.ass	contains the data summary created by the !SUM qualifier (see page 68).
.dbr/.dpr/.spr	contains the data and residuals in a binary form for further analysis (see !RESIDUALS, Table 5.5).
.veo	holds the equation order to speed up re-running big jobs when the model is unchanged. This binary file is of no use to the user.
.vll	holds factor level names when data/residuals are saved in binary form. See !SAVE on page 81.
.vrb	contains the estimates of the fixed effects and their variance if !VRB qualifier specified.
.vvp	contains the approximate variances of the variance parameters. It is designed to be read back for calculating functions of the variance parameters (see VPREDICT in chapter 13.
.was	*basename*.was is open while ASReml is running and deleted when it finishes. It will normally be invisible to the user unless the job crashes. It is used by ASReml-W to tell when the job finishes.
.xml	contains key information from the .asr, .pvs and .res file in a form easier for computers to parse.

An ASReml run generates many files and the .sln and .yht files, in particular, are often quite large and could fill up your disk space. You should therefore regularly tidy your working directories, maybe just keeping the .as, .asr, .rsv and .pvs files.

14.2 An example

In this chapter the ASReml output files are discussed with reference to a two-dimensional separable autoregressive spatial analysis of the NIN field trial data, see model **3b** on page 120 of Chapter 7 for details. The ASReml command file for this analysis is presented to the right. Recall that this model specifies a separable autoregressive correlation structure for residual or plot errors that is the direct product of an autoregressive correlation matrix of order 22 for rows and an autoregressive correlation matrix of order 11 for columns.

```
NIN Alliance Trial 1989
 variety !A
 id
 pid
 raw
 repl 4
 nloc
 yield
 lat
 long
 row 22
 column 11
nin89a.asd !skip 1 !DISPLAY 15
tabulate yield ~ variety
yield ~ mu variety !f mv
residual ar1(row).ar1(column)
predict variety
```

14.3 Key output files

The key ASReml output files are the `.asr`, `.sln` and `.yht` files.

14.3.1 The `.asr` file

This file contains

- an announcements box (outlined in asterisks) containing current messages,
- a summary of the data for the user to confirm the data file has been interpreted correctly and to review the basic structure of the data and validate the specification of the model,
- the iteration sequence of REML loglikelihood values to check convergence,
- a summary of the variance parameters:
  - The Gamma column reports the actual parameter fitted,
  - the Sigma column reports the gamma converted to a variance scale if appropriate,
  - Sigma/SE is the ratio of the component relative to the square root of the diagonal element of the inverse of the average information matrix Warning Sigma/SE should not be used for formal testing,
  - The % shows the percentage change in the parameter at the last iteration,
  - use `VPREDICT` (see Chapter 13) to calculate meaningful functions of the variance components,
- a table of Wald F statistics for testing fixed effects. (Section 6.11). The table contains the

numerator degrees of freedom for the terms and 'incremental' F-statistics for approximate testing of effects. It may also contain denominator degrees of freedon, a 'conditional' Wald F statistic and a significance probability.

- estimated effects, their standard errors and t values for equations in the DENSE portion of the SSP matrix are reported if !BRIEF -1 is invoked; the T-prev column tests difference between successive coefficients in the same factor.

The reported log-likelihood value may be positive or negative and typically excludes some constants from its calculation. It is sometimes reported relative to an offset (when its magnitude exceeds 10000); any offset is reported in the .asr file. Twice the difference in the likelihoods for two models is commonly used as the basis for a likelihood ratio test (see page 16). This is not valid for generalised linear mixed models as the reported LogL does not include components relating to the reweighting. Furthermore, it is not appropriate if the fixed effects in the model have changed. In particular, if fixed effects are fitted in the sparse equations, the order of fitting may change with a change in the fitted variance structure resulting in non comparable likelihoods even though the fixed terms in the model have not changed. The iteration sequence terminates when the maximum iterations (see !MAXIT on page 68) has been reached or successive LogL values are less than $0.002i$ apart.

The following is a copy of nin89a.asr.

```
ASReml 4.0 [01 Jan 2013] NIN Alliance Trial 1989 version & title
  Build ki [07 Jan 2014]   64 bit date
29 Jan 2014 09:34:34.315     32 Mbyte Windows x64  nin89a workspace

Licensed to: VSNi (Robin Thompson)    3
*****************************************************************
* Contact support@asreml.co.uk for licensing and support        *
*********************************************************** ARG *
Folder: D:\latest\Data\examples4\arg\Manex4f
variety !A
QUALIFIERS: !SKIP 1 !DISPLAY 15
QUALIFIER: !DOPART    1 is active
Reading nin89aug.asd  FREE FORMAT skipping     1 lines

Univariate analysis of yield
Summary of 242 records retained of 242 read data summary


 Model term          Size #miss #zero   MinNon0    Mean      MaxNon0  StndDevn
  1 variety            56     0     0      1    26.4545         56
  2 id                        0     0 1.0000      26.45      56.00      17.18
  3 pid                      18     0  1101.      2628.      4156.      1121.
  4 raw                      18     0  21.00      510.5      840.0      149.0
  5 repl                4     0     0      1     2.4132          4
  6 nloc                      0     0  4.000      4.000      4.000      0.000
  7 yield          Variate   18     0  1.050      25.53      42.00      7.450
  8 lat                       0     0  4.300      25.80      47.30      13.63
  9 long                      0     0  1.200      13.80      26.40      7.629
 10 row                22     0     0      1    11.5000         22
 11 column             11     0     0      1     6.0000         11
 12 mu                               1
 13 mv_estimates                    18
ar1v(row) in ar1(row).ar1(column) has size 22, parameters:   5   5
ar1(column) in ar1(row).ar1(column) has size 11, parameters:   6   6
 ar1(row).ar1(column)                [   4:   6] initialized.
Sorting Section  1:  22 rows by  11 columns
Forming       75 equations:  57 dense.
Initial updates will be shrunk by factor    0.316

 Notice: Specify !SIGMAP to allow the Sigma parameterisation
Notice:      1 singularities detected in design matrix.  iterations
  1 LogL=-449.818     S2=  49.775        168 df    1.0000      0.1000E00 0.1000E+00+
  2 LogL=-424.315     S2=  40.233        168 df    1.0000      0.2937      0.2323
  3 LogL=-405.419     S2=  38.922        168 df    1.0000      0.4813      0.3587
  4 LogL=-399.552     S2=  45.601        168 df    1.0000      0.6156      0.4398
  5 LogL=-399.336     S2=  47.986        168 df    1.0000      0.6456      0.4417
  6 LogL=-399.325     S2=  48.546        168 df    1.0000      0.6530      0.4391
  7 LogL=-399.324     S2=  48.672        168 df    1.0000      0.6549      0.4380
  8 LogL=-399.324     S2=  48.703        168 df    1.0000      0.6554      0.4376
Final parameter values                           1.0000      0.6555      0.4375

         - - - Results from analysis of yield - - -
Akaike Information Criterion      804.65 (assuming 3 parameters).
Bayesian Information Criterion    814.02

Model_Term                              Gamma         Sigma   Sigma/SE   % C
ar1(row).ar1(column)           242  effects
Residual                  SCA_V  242   1.000000        48.7026       6.81   0 P  parameter
row                        AR_R    1   0.655480      0.655480      11.63   0 P  estimates
column                     AR_R    1   0.437505      0.437505       5.43   0 P

                                 Wald F statistics
    Source of Variation          NumDF     DenDF     F-inc            P-inc  testing
 12 mu                               1      25.0    331.93            <.001  fixed
  1 variety                         55     110.8      2.22            <.001  effects
Notice: The DenDF values are calculated ignoring fixed/boundary/singular
            variance parameters using algebraic derivatives.
 13 mv_estimates                          18 effects fitted
       6 possible outliers: in section  11 (see .res file)
Finished: 29 Jan 2014 09:34:34.861   LogL Converged
```

Following is a table of Wald F statistics augmented with a portion of Regression Screen output. The qualifier was `!SCREEN 3 !SMX 3`.

```
Model_Term                              Gamma         Sigma    Sigma/SE   % C
idsize                  IDV_V    92  0.581102      0.136683        3.31    0 P
expt.idsize             IDV_V   828  0.121231      0.285153E-01    1.12    0 P
idv(units)                      504 effects
Residual                SCA_V   504  1.000000      0.235214       12.70    0 P

                                 Wald F statistics
    Source of Variation          NumDF     DenDF_con F_inc     F_con M P_con
113 mu                               1      72.4 65452.25 56223.68 . <.001
  2 expt                             6      37.5      5.27      0.64 A 0.695
```

```
   4 type                          4      63.8    22.95     3.01 A 0.024
 114 expt.type                    10      79.3     1.31     0.93 B 0.508
  23 x20                           1      55.1     4.33     2.37 B 0.130
  24 x21                           1      63.3     1.91     0.87 B 0.355
  25 x23                           1      68.3    23.93     0.11 B 0.745

  26 x39                           1      79.7     1.85     0.35 B 0.556
  27 x48                           1      69.9     1.58     2.08 B 0.154
  28 x59                           1      49.7     1.41     0.08 B 0.779
  29 x60                           1      59.6     1.46     0.42 B 0.518
  30 x61                           1      64.0     1.11     0.04 B 0.838

  31 x62                           1      61.8     2.18     0.09 B 0.770
  32 x64                           1      55.6    31.48     4.50 B 0.038
  33 x65                           1      57.8     4.72     6.12 B 0.016
  34 x66                           1      58.5     1.13     0.03 B 0.872
  35 x70                           1      59.3     1.71     1.40 B 0.242

  36 x71                           1      64.4     0.08     0.01 B 0.929
  37 x73                           1      59.0     1.79     3.01 B 0.088
  38 x75                           1      59.9     0.04     0.26 B 0.613
  39 x91                           1      63.8     1.44     1.44 B 0.234
 Notice: The DenDF values are calculated ignoring fixed/boundary/singular
             variance parameters using empirical derivatives.

 129 mv_estimates                        9 effects fitted
   9 idsize                             92 effects fitted (       7 are zero)
 115 expt.idsize                       828 effects fitted (     672 are zero)
 127 at(expt,6).type.idsize.meth         9 effects fitted (+    2199 singular)
 128 at(expt,7).type.idsize.meth        10 effects fitted (+    2198 singular)
```

```
LINE REGRESSION      RESIDUAL              ADJUSTED  FACTORS INCLUDED
 NO DF SUMSQUARES     DF MEANSQU R-SQUARED R-SQUARED 39 38 37 36 35 34 33 32 31 30 29 28 27 26 25 24 23
  1  3  0.1113D+02  452  0.2460  0.09098   0.08495   1  1  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0
                                 *****     *****
  2  3  0.1180D+02  452  0.2445  0.09648   0.09049   1  0  1  1  0  0  0  0  0  0  0  0  0  0  0  0  0
                                 *****     *****
  3  3  0.1843D+01  452  0.2666  0.01507   0.00853   0  1  1  1  0  0  0  0  0  0  0  0  0  0  0  0  0
  4  3  0.1095D+02  452  0.2464  0.08957   0.08353   1  1  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0
  5  3  0.1271D+02  452  0.2425  0.10390   0.09795   1  0  0  1  1  0  0  0  0  0  0  0  0  0  0  0  0
                                 *****     *****
  6  3  0.9291D+01  452  0.2501  0.07594   0.06981   0  1  0  1  1  0  0  0  0  0  0  0  0  0  0  0  0
  7  3  0.9362D+01  452  0.2499  0.07652   0.07039   0  0  1  1  1  0  0  0  0  0  0  0  0  0  0  0  0
  8  3  0.1357D+02  452  0.2406  0.11091   0.10501   1  0  1  0  1  0  0  0  0  0  0  0  0  0  0  0  0
                                 *****     *****
  9  3  0.9404D+01  452  0.2498  0.07687   0.07074   0  1  1  0  1  0  0  0  0  0  0  0  0  0  0  0  0
 10  3  0.1266D+02  452  0.2426  0.10350   0.09755   1  1  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0
 11  3  0.1261D+02  452  0.2427  0.10313   0.09717   1  0  0  0  1  1  0  0  0  0  0  0  0  0  0  0  0
 12  3  0.9672D+01  452  0.2492  0.07906   0.07295   0  1  0  0  1  1  0  0  0  0  0  0  0  0  0  0  0
 13  3  0.9579D+01  452  0.2494  0.07830   0.07218   0  0  1  0  1  1  0  0  0  0  0  0  0  0  0  0  0
 14  3  0.9540D+01  452  0.2495  0.07797   0.07185   0  0  0  1  1  1  0  0  0  0  0  0  0  0  0  0  0
 15  3  0.1089D+02  452  0.2465  0.08907   0.08302   1  0  0  1  0  1  0  0  0  0  0  0  0  0  0  0  0
 16  3  0.2917D+01  452  0.2642  0.02384   0.01736   0  1  0  1  0  1  0  0  0  0  0  0  0  0  0  0  0
 17  3  0.2248D+01  452  0.2657  0.01838   0.01187   0  0  1  1  0  1  0  0  0  0  0  0  0  0  0  0  0
 18  3  0.1111D+02  452  0.2460  0.09088   0.08484   1  0  1  0  0  1  0  0  0  0  0  0  0  0  0  0  0
 19  3  0.1746D+01  452  0.2668  0.01427   0.00773   0  1  1  0  0  1  0  0  0  0  0  0  0  0  0  0  0
 20  3  0.1030D+02  452  0.2478  0.08423   0.07815   1  1  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0
 21  3  0.1279D+02  452  0.2423  0.10454   0.09860   1  0  0  0  0  1  1  0  0  0  0  0  0  0  0  0  0
 22  3  0.8086D+01  452  0.2527  0.06609   0.05989   0  1  0  0  0  1  1  0  0  0  0  0  0  0  0  0  0
 23  3  0.7437D+01  452  0.2542  0.06079   0.05456   0  0  1  0  0  1  1  0  0  0  0  0  0  0  0  0  0
 24  3  0.1071D+02  452  0.2469  0.08755   0.08149   0  0  0  1  0  1  1  0  0  0  0  0  0  0  0  0  0
 25  3  0.1370D+02  452  0.2403  0.11200   0.10611   0  0  0  0  1  1  1  0  0  0  0  0  0  0  0  0  0
                                 *****     *****
 26  3  0.1511D+02  452  0.2372  0.12351   0.11770   1  0  0  0  1  0  1  0  0  0  0  0  0  0  0  0  0
                                 *****     *****
 27  3  0.1353D+02  452  0.2407  0.11064   0.10473   0  1  0  0  1  0  1  0  0  0  0  0  0  0  0  0  0
...
680  3  0.1057D+02  452  0.2472  0.08641   0.08035   1  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1
```

The primary tables reported in the `.asr` file are now also written in XML format to a `.xml` file. The intended use of this file is by programs written to parse Asreml output. The information contained in the `.xml` file includes start and finish times, the data summary, the iteration sequence summary, the summary of estimated variance structure parameters and the Wald F statistics. Developers are advised to parse the `.xml` file in redeveloping code to handle the changes with the new release.

14.3.2 The .sln file

The .sln file contains estimates of the fixed and random effects with their standard errors in an array with four columns ordered as

factor_name level estimate standard_error

Note that the error presented for the estimate of a random effect is the square root of the prediction error variance. In a genetic context for example where a relationship matrix $\boldsymbol{A}$ is involved, the accuracy is $\sqrt{(1 - \frac{s_i^2}{(1+f_i)\sigma_A^2})}$ where s_i is the standard error reported with the BLUP (u_i) for the ith individual, f_i is the inbreeding coefficient reported when !DIAG qualifier is given on a pedigree file line, $1 + f_i$ is the diagonal element of $\boldsymbol{A}$ and σ_A^2 is the genetic variance. The .sln file can easily be read into a GENSTAT spreadsheet or an S-PLUS data frame. Below is a truncated copy of nin89a.sln. Note that

- the order of some terms may differ from the order in which those terms were specified in the model statement,

- the missing value estimates appear at the end of the file in this example.

- the format of the file can be changed by specifying the !SLNFORM qualifier. In particular, more significant digits will be reported.

- use of the !OUTLIER qualifier will generate extra columns containing the outlier statistics described on page 17.

```
Model_Term                Level                  Effect     seEffect
 variety                  LANCER                  0.000       0.000 variety estimates
 variety                  BRULE                   2.984       2.841
 variety                  REDLAND                 4.706       2.977
 variety                  CODY                  -0.3158       2.961
 variety                  ARAPAHOE                2.954       2.727
 - - -
 variety                  NE87615                 1.033       2.934
 variety                  NE87619                 5.937       2.849
 variety                  NE87627                -4.378       2.997
 mu                                 1             24.09       2.465 intercept
 mv_estimates                       1             21.91       6.731 missing value estimates
 mv_estimates                       2             23.22       6.723
 mv_estimates                       3             22.52       6.711
 mv_estimates                       4             23.49       6.678
 mv_estimates                       5             22.27       6.700
 mv_estimates                       6             24.47       6.709
 mv_estimates                       7             20.14       6.699
 mv_estimates                       8             25.01       6.693
 mv_estimates                9                 24.29       6.678
 mv_estimates               10                 26.30       6.660
 mv_estimates               11                 24.99       6.592
 mv_estimates               12                 27.78       6.493
 mv_estimates               13                 25.39       6.305
 mv_estimates               14                 26.81       5.900
 mv_estimates               15                 29.07       4.906
 mv_estimates               16                 23.97       4.577
 mv_estimates               17                 24.27       4.618
 mv_estimates               18                 29.82       4.532
```

14.3.3 The .yht file

The .yht file contains the predicted values of the data in the original order (this is not changed by supplying row/column order in spatial analyses), the residuals and the diagonal elements of the hat matrix. Figure 14.1 shows the residuals plotted against the fitted values (Yhat) and a line printer version of this figure is written to the .res file. Where an observation is missing, the residual, missing values predicted value and Hat value are also declared missing. The missing value estimates with standard errors are reported in the .sln file.

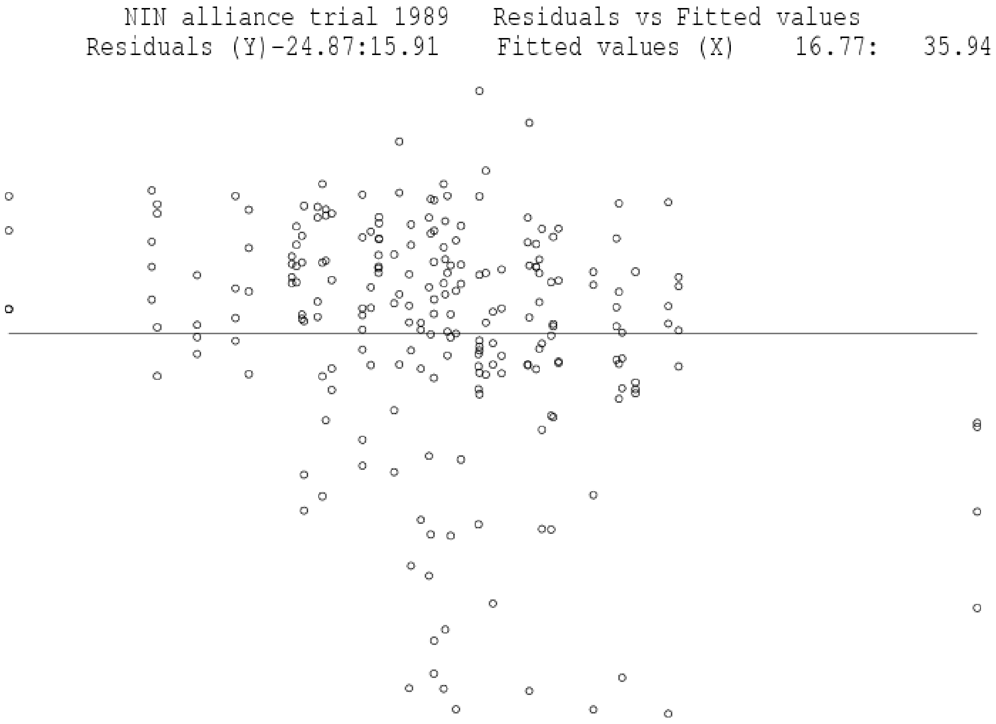

Figure 14.1: Residual versus Fitted values

This is part of nin89a.yht. Note that the values corresponding to the missing data (first 15 records) are all -0.1000E-36 which is the internal value used for missing values.

```
 Record         Yhat    Residual          Hat
       1    -0.10000E-36 -0.1000E-36 -0.1000E-36
       2    -0.10000E-36 -0.1000E-36 -0.1000E-36
       3    -0.10000E-36 -0.1000E-36 -0.1000E-36
       4    -0.10000E-36 -0.1000E-36 -0.1000E-36
       - - -
      15    -0.10000E-36 -0.1000E-36 -0.1000E-36
      16      24.089       5.161       6.075
      17      27.073       4.477       6.223
      18      28.795       6.255       6.283
      19      23.773       6.327       6.236
      20      27.043       6.007       5.963
       - - -
     239      21.522       8.128       6.314
     240      24.696       1.854       6.114
     241      25.452      0.1480       6.159
     242      22.464       4.436       6.605
```

14.4 Other ASReml output files

14.4.1 The .aov file

This file reports details of the calculation of Wald F statistics, particularly as relating to the conditional Wald F statistics (not computed in this demonstration). In the following table relating to the incremental Wald F statistic, the columns are

- model term
- columns in design matrix
- numerator degrees of freedom
- simple Wald F statistic
- Wald F statistic scaled by λ
- λ as defined in Kenward & Roger.

```
denominator degrees of freedom
 Source                    Size NumDF   F-value  Lambda*F    Lambda   DenDF
 mu                           1     1  331.9252  331.9252    1.0000   25.0143
 variety                     56    55    2.2257    2.2245    0.9995  110.8370
```

A more useful example is obtained by adding a linear nitrogen contrast to the oats example (Section 16.2).

The basic design is six replicates of three whole plots to which variety was randomised, and four subplots which received 4 rates of nitrogen. A !CONTRAST qualifier defines the model term linNitr as the linear covariate representing ntrogen applied. Fitting this before the model term nitrogen means that this latter term represents lack of fit from a linear response.

```
Split plot analysis - oat
  blocks *
  nitrogen !A
  subplots
  variety !A
  wplots *
  yield
oats.asd !skip 2
!CONTRAST linNitr nitrogen .6 .4 .2 0
!FCON
yield ~ mu variety linNitr nitrogen,
variety.linNitr variety.nitrogen,
  !r idv(blocks) idv(blocks.wplots)
residual idv(units)
```

The !FCON qualifier requests conditional Wald F statistics. As this is a small example, denominator degrees of freedom are reported by default. An extract from the .asr file is followed by the contents of the .aov file.

```
          - - - Results from analysis of yield - - -
 Akaike Information Criterion      415.10 (assuming 3 parameters).
 Bayesian Information Criterion    421.38

          Approximate stratum variance decomposition
 Stratum     Degrees-Freedom   Variance      Component Coefficients
 idv(blocks)            5.00    3175.06        12.0      4.0      1.0
 idv(blocks.wplots     10.00    601.331         0.0      4.0      1.0
 Residual Variance     45.00    177.083         0.0      0.0      1.0

 Model_Term                             Gamma         Sigma   Sigma/SE   % C
 idv(blocks)             IDV_V    6   1.21116       214.477       1.27    0 P
 idv(blocks.wplots)      IDV_V   18  0.598937       106.062       1.56    0 P
 idv(units)                      72 effects
 Residual                SCA_V   72  1.000000       177.083       4.74    0 P

                                   Wald F statistics
     Source of Variation           NumDF     DenDF_con F-inc     F-con M P-con
   8 mu                                1         6.0   245.14   138.14 . <.001
   4 variety                           2        10.0     1.49     1.49 A 0.272
   7 linNitr                           1        45.0   110.32   110.32 a <.001
   2 nitrogen                          2        45.0     1.37     1.37 A 0.265
   9 variety.linNitr                   2        45.0     0.48     0.48 b 0.625
  10 variety.nitrogen                  4        45.0     0.22     0.22 B 0.928
```

The analysis shows that there is a significant linear response to nitrogen level but the lack of fit term and the interactions with variety are not significant. In this example, the conditional Wald F statistic is the same as the incremental one because the contrast must appear before the lack-of-fit and the main effect before the interaction and otherwise it is a balanced analysis.

The first part of the `.aov` file, the `FMAP` table only appears if the job is run in DEBUG mode. There is a line for each model term showing the number of non-singular effects in the terms before the current term is absorbed. For example, `variety.nitrogen` initially has 12 degrees of freedom (non-singular effects). `mu` takes 1, `variety` then takes 2, `linNitr` takes 1, `nitrogen` takes 2, `variety.linNitr` takes 2 and there are four degrees of freedom left. This information is used to make sure that the conditional Wald F statistic does not contradict marginality principles.

The next table indicates the details of the conditional Wald F statistic. The conditional Wald F statistic is based in the reduction in Sums of Squares from dropping the particular term (indicated by `*`) from the model also including the terms indicated by `I`, `C` and `c`.

The next two tables, based on incremental and conditional sums of squares report the model term, the number of effects in the term, the (numerator) degrees of freedom, the Wald F statistic, an adjusted Wald F statistic scaled by a constant reported in the next column and finally the computed denominator degrees of freedom. The scaling constant is discussed by Kenward and Roger (1997).

```
 Table showing the reduction in the numerator degrees of freedom
                  for each term as higher terms are absorbed.
 Model Term              6  5  4  3  2  1
 1 mu                   12  3  4  1  3  1
 2 variety              11  3  3  1  2
 3 LinNitr               9  3  3  1
 4 nitrogen              8  2  2
 5 variety.LinNitr       6  2
 6 variety.nitrogen      4

                 Marginality pattern for F-con calculation
                          -- Model terms --
Model Term           DF   1  2  3  4  5  6

 1 mu                 1   *  .  C  .  C  .
 2 variety            2   I  *  C  C  .  .
 3 LinNitr            1   I  I  *  .  .  .
 4 nitrogen           2   I  I  I  *  .  .
 5 variety.LinNitr    2   I  I  I  I  *  .

 6 variety.nitrogen   4   I  I  I  I  I  *
          Model codes:    b  A  a  A  b  B
F-inc tests the additional variation explained when the term (*)
      is added to a model consisting of the I terms.
F-con tests the additional variation explained when the term (*)
      is added to a model consisting of the I and C/c terms.
      The . terms are ignored for both F-inc and F-con tests.


    Incremental  F statistics - calculation of Denominator degrees of freedom
Source                Size NumDF   F-value  Lambda*F     Lambda   DenDF
mu                       1     1  245.1409  245.1409     1.0000   5.0000
variety                  3     2    1.4853    1.4853     1.0000  10.0000
```

```
LinNitr                  1     1  110.3232  110.3232    1.0000   45.0000
nitrogen                 4     2    1.3669    1.3669    1.0000   45.0000
variety.LinNitr          3     2    0.4753    0.4753    1.0000   45.0000
variety.nitrogen        12     4    0.2166    0.2166    1.0000   45.0000

    Conditional  F statistics - calculation of Denominator degrees of freedom
Source                Size NumDF   F-value  Lambda*F    Lambda   DenDF
mu                       1     1  138.1360  138.1360    1.0000    6.0475
variety                  3     2    1.4853    1.4853    1.0000   10.0000
LinNitr                  1     1  110.3232  110.3232    1.0000   45.0000
nitrogen                 4     2    1.3669    1.3669    1.0000   45.0000
variety.LinNitr          3     2    0.4753    0.4753    1.0000   45.0000
variety.nitrogen        12     4    0.2166    0.2166    1.0000   45.0000
```

14.4.2 The .asl file

The .asl file is primarily used for low-level debugging. It is produced when the !LOGFILE qualifier is specified and contains lowlevel debugging information information when the !DEBUG qualifier is also given.

However, when a job running on a Unix system crashes with a Segmentation fault, the output buffers are not flushed so the output files do not reflect the latest program output. In this case, use the Unix script screen.log command before running ASReml with the !DEBUG qualifier but without the !LOGFILE qualifier, to capture all the debugging information in the file screen.log.

The debug information pertains particularly to the first iteration and includes timing information reported in lines beginning >>>> >>>> >>>>. These lines also mark progress through the iteration.

14.4.3 The .dpr file

The .dpr file contains the data and residuals from the analysis in double precision binary form. The file is produced when the !RES qualifier (Table 4.3) is invoked. The file could be renamed with filename extension .dbl and used for input to another run of ASReml. Alternatively, it could be used by another Fortran program or package. Factors will have level codes if they were coded using !A or !I. All the data from the run plus an extra column of residuals is in the file. Records omitted from the analysis are omitted from the file.

14.4.4 The .msv file

The .msv file contains the variance parameters from the most recent iteration of a model in a form that is relatively easy to edit if the values need to be reset. The file is read when !MSV or !CONTINUE 3 is specified. This is nin89a.msv:

```
# This .msv file is a mechanism for resetting initial parameter values
# by changing the values here and rerunning the job with !CONTINUE 3.
# You may not change values in the first 3 fields
#                    or RP fields where RP_GN is negative.

# Fields are:
# GN, Term, Type, PSpace, Initial_value, RP_GN, RP_scale.

   4, "Variance 1", V, P,   1.00000000    ,    4,    1
   5, "ar1(row).ar1(column);ar1v(row)_1", R, P,   0.65547976    ,    5,    1
   6, "ar1(row).ar1(column);ar1(column)_1", R, P,   0.43750453    ,    6,    1

# Valid values for Pspace are F, P, U and maybe Z.

# RP_GN and RP_scale define simple parameter relationships;
# RP_GN links related parameters by the first GN number;
# RP_scale must be 1.0 for the first parameter in the set and
# otherwise specifies the size relative to the first parameter.
# Multivalue RP_scale parameters may not be altered here.

# Notice that this file is overwritten if not being read.
```

14.4.5 The .pvc file

The .pvc file contains functions of the variance components produced by running a .pin file on the results of an ASReml run as described in Chapter 13. The .pin and .pvc files for a half-sib analysis of the Coopworth data are presented in Section 16.10.

14.4.6 The .pvs file

The .pvs file contains the predicted values formed when a predict statement is included in the job. Below is an edited version of nin89a.pvs. See Section 3.6 for the .pvs file for the simple RCB analysis of the NIN data considered in that chapter.

```
NIN Alliance Trial 1989                                    03 Feb 2014 06:23:03 title line
                                                            nin89a

 Ecode is E for Estimable, * for Not Estimable

 Warning: mv_estimates         is ignored for prediction
 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.

 ---- ---- ---- ---- ---- ----    1  ---- ---- ---- ---- ---- ----
 Predicted values of yield

 variety         Predicted_Value Standard_Error Ecode predicted variety means
 LANCER                  24.0891         2.4648 E
 BRULE                   27.0731         2.4946 E
 REDLAND                 28.7953         2.5066 E
 CODY                    23.7733         2.4973 E
 ARAPAHOE                27.0429         2.4420 E
 NE83404                 25.7199         2.4426 E
 NE83406                 25.3793         2.5030 E
 NE83407                 24.3981         2.6882 E
 CENTURA                 26.3531         2.4765 E
 SCOUT66                 29.1741         2.4363 E
 - - -
 NE87615                 25.1218         2.4436 E
 NE87619                 30.0261         2.4669 E
 NE87627                 19.7108         2.4836 E
 SED: Overall Standard Error of Difference    2.925 SED summary
```

14.4.7 The .res file

The .res file contains miscellaneous supplementary information including

- a list of unique values of x formed by using the fac() model term,
- a list of unique (x, y) combinations formed by using the fac(x,y) model term,
- legandre polynomials produced by leg() model term,
- orthogonal polynomials produced by pol() model term,
- the design matrix formed for the spl() model term,

- predicted values of the curvature component of cubic smoothing splines,

- the empirical variance-covariance matrix based on the BLUPs when a $\boldsymbol{\Sigma} \otimes \boldsymbol{I}$ or $\boldsymbol{I} \otimes \boldsymbol{\Sigma}$ structure is used; this may be used to obtain starting values for another run of ASReml,

- a table showing the variance components for each iteration,

- a figure and table showing the variance partitioning for any XFA structures fitted,

- some statistics derived from the residuals from two-dimensional data (multivariate, repeated measures or spatial)
  - the residuals from a spatial analysis will have the `units` part added to them (defined as the combined residual) unless the data records were sorted (within ASReml) in which case the `units` and the correlated residuals are in different orders (data file order and field order respectively),

  - the residuals are printed in the `.yht` file but the statistics in the `.res` file are calculated from the combined residual,

  - the `Covariance/Variance/Correlation (C/V/C)` matrix calculated directly from the residuals; it contains the covariance below the diagonals, the variances on the diagonal and the correlations above the diagonal:

    The fitted matrix is the same as is reported in the `.asr` file and if the Logl has converged is the one you would report. The BLUPs matrix is calculated from the BLUPs and is provided so it can be used as starting values when a simple initial model has been used and you are wanting to attempt to fit a full unstructured matrix. For computational reasons, it pertains to the parameters and so may differ from the parameter values generated by the last iteration. The BLUPs matrix may look quite different from the fitted matrix because BLUPs are shrunken phenotypes. The BLUPs matrix retains much of the character of the phenotypes; the rescaled has the variance from the fitted and the covariance from the BLUPs and might be more suitable as an initial matrix if the variances have been estimated. The BLUPs and rescaled matrices should not be reported.

  - relevant portions of the estimated variance matrix for each term for which an R structure or a G structure has been associated,

- a variogram and spatial correlations for spatial analysis; the spatial correlations are based on distance between data points (see Gilmour *et al.*, 1997),

- the slope of the *log(absolute residual)* on *log(predicted value)* for assessing possible mean-variance relationships and the location of large residuals. For example,

  ```
  SLOPES FOR LOG(ABS(RES)) ON LOG(PV) for section 1
  0.99 2.01 4.34
  ```

  produced from a trivariate analysis reports the slopes. A slope of b suggests that y^{1-b}

might have less mean variance relationship. If there is no mean variance relation, a slope of zero is expected. A slope of $\frac{1}{2}$ suggests a SQRT transformation might resolve the dependence; a slope of 1 means a LOG transformation might be appropriate. So, for the 3 traits, $log(y_1)$, y_2^{-1} and y_3^{-3} are indicated. This diagnostic strategy works better when based on grouped data regressing *log(standard deviation)* on *log(mean)*. Also,

```
STND RES 16 -2.35 6.58 5.64
```

indicates that for the 16th data record, the residuals are -2.35, 6.58 and 5.64 times the respective standard deviations. The standard deviation used in this test is calculated directly from the residuals rather than from the analysis. They are intended to flag the records with large residuals rather than to precisely quantify their relative size. They are not studentised residuals and are generally not relevant when the user has fitted heterogeneous variances.

```
=== === === === Residual statistics for nin891.asr === === === ===

Convergence sequence of variance parameters

Iteration    1          2          3          4          5          6
LogL   -449.818   -424.315   -405.419   -399.552   -399.336   -399.325
Change %   177        216        201         51         13          3
Adjusted     0          0          0          0          0          0
StepSz     0.316      0.562      1.000      1.000      1.000      1.000
  5 R   0.100000   0.293737   0.481321   0.615630   0.645607   0.653013    1.1
  6 R   0.100000   0.232335   0.358720   0.439779   0.441733   0.439143   -0.7
         Trace of W(W'R^W+G^)^W'  1376.1714

Plot of Residuals [  -24.8729   15.9146] vs Fitted values [   16.7728   35.9349] _RvE11
    -----------------------------1------------------------------
    .                               1                          .
    .                       1                                  .
    .                  1      1  1                             .
    1       12    2   1211  1 21 1       1  1                  .
    1                112 15 1 311   121  1                     .
    .       1     1  312  111 221   3                          .
    .       1  1  1  4  1 4 1 22121 411 2 1  2                 .
    2       1     1   11 1112 23 11 1    2  1                  .
    .        1 2  1   21 2  1213 1  13   2  11                 .
    -----------1--1------1--2--1-61-212--3----------------------
    .        1    1    11 1  11  41 2    12  1                 .
    .                   1  1     1       11                    .
    .                  1             3                         2
    .                    1    1 1                              .
    .                 1  1 1                                   .
    .                 11                1                      1
    .                        111 1   2                         .
    .                                                          .
    .                       1 1                                .
    .                            1                             1
    .                          1                               .
    .                         1                                .
    .                       1 2          1                     .
    ---------------------------1----1---1---1-------------------
SLOPES FOR LOG(ABS(RES)) on LOG(PV) for Section  11
  0.15
SLOPES FOR LOG(SDi) on LOG(PVBari) for Section  11
  1.37
                                       *
                                       *
                              *   ** ***
                              *   ** *** *
                              ** *** *** *** *
                              ********** *****
                            ******************
```

```
* *             *     *  ** ******************
* **  ** ** ** ***** ** ************************  **  *

Min Mean Max  -24.873     0.27959      15.915     omitting    18 zeros


Spatial diagnostic statistics of Residuals      22   11
 Residual Plot   and    Autocorrelations
  <LOo- +xXH>             [se   0.077]
 |               ++xxx+X|
 |--- - O+ + x +x+x>+X  |
 |o - -- +   +++xxx++X++|
 |+      + + +x- +xxx+++|
 |   o -- ++  +- xx+xHxx|
 |-+xxx+xXx +++x xX  ++x|
 |-++ o- +XxxXXx-xXX +++|
 |ooL<Oo --++x x+xXx+x+H|
 |<<<<<OO-- xX+  -x ++--|
 |<O<<LLLoo -  -o-+-+  +|
 |L<<<<O-OL-o  -++x x+ +|
  1  0.28  0.38  0.50  0.65  0.77  1.00  0.77  0.65  0.50  0.38  0.28
  2  0.17  0.27  0.39  0.51  0.56  0.64  0.56  0.50  0.40  0.32  0.26
  3  0.05  0.11  0.19  0.28  0.35  0.42  0.40  0.35  0.30  0.24  0.19

  Residuals [Percentage of sigma =    6.979    ]
    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0   74   64   90   91   86   65  141
  -72  -29  -52  -20  -61   11 -132   26    0   63   15   99    9   37   84   48  110  228   49  131  -20    9
  -87    1  -32  -14  -26  -30   -3   37   -6    4   23   32   44   46  109   97   83   67   68  141   69   40
   44   11    0    3    6    0   21   41  -15   51   25   32  120  -33   10   58  117  113  109   63   57   25
   18   18   -2  -84  -19  -51  -45   18   30   56   -9  -12   53  -41    7   99  123   47  119  181  101  104
  -40   29   87  103   81   61   81  130   94   10   55   53   55  106   15  109  153   23    0   50   66  111
  -29   75   43  -24  -90  -37  -23   64  130   84  122  129  127   90  -38   91  133  126  -16   57   30   70
  -99 -114 -218 -332 -174  -77  -19  -38  -29   58   63   88    4  124   49  101  129  113   45   92   70  198
 -257 -333 -352 -319 -253 -166 -152  -52  -28    0   97  135   67   16   -9  -36   96   24   62   48  -27  -29
 -227 -167 -356 -335 -183 -179 -189 -118 -124   14  -52   19   -7  -56  -81  -33   63  -40   57  -15   24   73
 -183 -277 -352 -323 -288 -151  -56 -130 -188  -29  -78    7   12  -30   39   57   89   -3  116   27    2   64

|                      |                      |                      |
|                      |                 ,    |                      |
|               , ,,, '|           ,  , ,  '  |              ,,,,,', |
|----------------'---'-|---,-,-',',-,'-'--'-,,|-,-,--,',,''''-------'|
|                      |,'' '                 |, ' ''                |
|                      |      ,               |                      |
|                      |                      |                      |

|                      |                      |                      |
|                      |                   ,  |                      |
|            '   '',   |               ,' ' ,,|  ,,, ,',    , ,'    ,|
|',,,,-,','''--,'---'''|,,,-,--,'',,'-,--'----|-'---'---,'''-,--'-''-|
|             '        |   , ''      '        |'                     |
|                      |                      |                      |
|                      |                      |                      |

|                      |                      |                      |
|                      |                     ,|                      |
| ,      ',''', ,''   ,|           , ' ,'' ,, |          ,',   ,     |
|--'----'----------,''-|------,--''-,-'---'---|-------------,,--'''--|
|'  ','       '       |,    , ''             |       ''      '    ''|
|                      | '  ,                 |     ,'               |
|                      |  '*                  |,***,                 |

|                      |                      |
|                      |                      |
|                     ,|                , '   |
|---------,-,,---'-','-|-----------,,-''-,-','|
|          '  ',' '    |      '  ',  '        |
| ,  ,,,''             |,    ' ',             |
|' **                  | ,**,                 |
Residual [section 11, column 8 (of 11), row 4 (of 22)] is -3.32 SD
```

```
 Residual [section 11, column 9 (of 11), row 2 (of 22)] is -3.33 SD
 Residual [section 11, column 9 (of 11), row 3 (of 22)] is -3.52 SD
 Residual [section 11, column 10 (of 11), row 3 (of 22)] is -3.56 SD
 Residual [section 11, column 10 (of 11), row 4 (of 22)] is -3.35 SD
 Residual [section 11, column 11 (of 11), row 3 (of 22)] is -3.52 SD
6 possible outliers in section 11: test value 23.0297999308
```

```
            Residuals [Percentage of sigma =   6.979    ]
    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0   74   64   90   91   86   65  141
  -72  -29  -52  -20  -61   11 -132   26    0   63   15   99    9   37   84   48  110  228   49  131  -20    9
  -87    1  -32  -14  -26  -30   -3   37   -6    4   23   32   44   46  109   97   83   67   68  141   69   40
   44   11    0    3    6    0   21   41  -15   51   25   32  120  -33   10   58  117  113  109   63   57   25
   18   18   -2  -84  -19  -51  -45   18   30   56   -9  -12   53  -41    7   99  123   47  119  181  101  104
  -40   29   87  103   81   61   81  130   94   10   55   53   55  106   15  109  153   23    0   50   66  111
  -29   75   43  -24  -90  -37  -23   64  130   84  122  129  126   90  -38   91  133  126  -16   57   30   70
  -99 -114 -218 -332 -174  -77  -19  -38  -29   58   63   88    4  124   49  101  129  113   45   92   70  198
 -257 -333 -352 -319 -253 -166 -152  -52  -28    0   97  135   67   16   -9  -36   96   24   62   48  -27  -29
 -227 -167 -356 -335 -183 -179 -189 -118 -124   14  -52   19   -7  -56  -81  -33   63  -40   57  -15   24   73
 -183 -277 -352 -323 -288 -151  -56 -130 -188  -29  -78    7   12  -30   39   57   89   -3  116   27    2   64
```

```
   |                      |                      |                      |
 |                      |                 ’    |                      |
 |               , ,,, ’|           ,  , ,  ’  |              ,,,,,’, |
 |----------------’---’-|---,-,-’,’,-,’-’--’-,,|-,-,--,’,,’’’’-------’|
 |                      |,’’ ’                 |, ’ ’’                |
 |                      |      ,               |                      |
 |                      |                      |                      |

 |                      |                      |                      |
 |                      |                   ,  |                      |
 |            ’   ’’,   |               ,’ ’ ,,|  ,,, ,’,    , ,’    ,|
 |’,,,,-,’,’’’--,’---’’’|,,,-,--,’’,,’-,--’----|-’---’---,’’’-,--’-’’-|
 |             ’        |   , ’’      ’        |’                     |
 |                      |                      |                      |
 |                      |                      |                      |

 |                      |                      |                      |
 |                      |                     ,|                      |
 | ,      ’,’’’, ,’’   ,|           , ’ ,’’ ,, |          ,’,   ,     |
 |--’----’----------,’’-|------,--’’-,-’---’---|-------------,,--’’’--|
 |’  ’,’’       ’       |,    , ’’             |       ’’      ’    ’’|
 |                      | ’  ,                 |     ,’               |
 |                      |  ’*                  |,***,                 |

 |                      |                      |
 |                      |                      |
 |                     ,|                , ’   |
 |---------,-,,---’-’,’-|-----------,,-’’-,-’,’|
 |          ’  ’,’ ’    |      ’  ’,  ’        |
 | ,  ,,,’’             |,    ’ ’,             |
 |’ **                  | ,**,                 |
 Residual [section  1, column   8 (  11), row   4 (  22)] is -3.32 SD
 Residual [section  1, column   9 (  11), row   2 (  22)] is -3.33 SD
 Residual [section  1, column   9 (  11), row   3 (  22)] is -3.52 SD
 Residual [section  1, column  10 (  11), row   3 (  22)] is -3.56 SD
 Residual [section  1, column  10 (  11), row   4 (  22)] is -3.35 SD
 Residual [section  1, column  11 (  11), row   3 (  22)] is -3.52 SD
           6  possible outliers in section 1 : test value   23.0311757288330
```

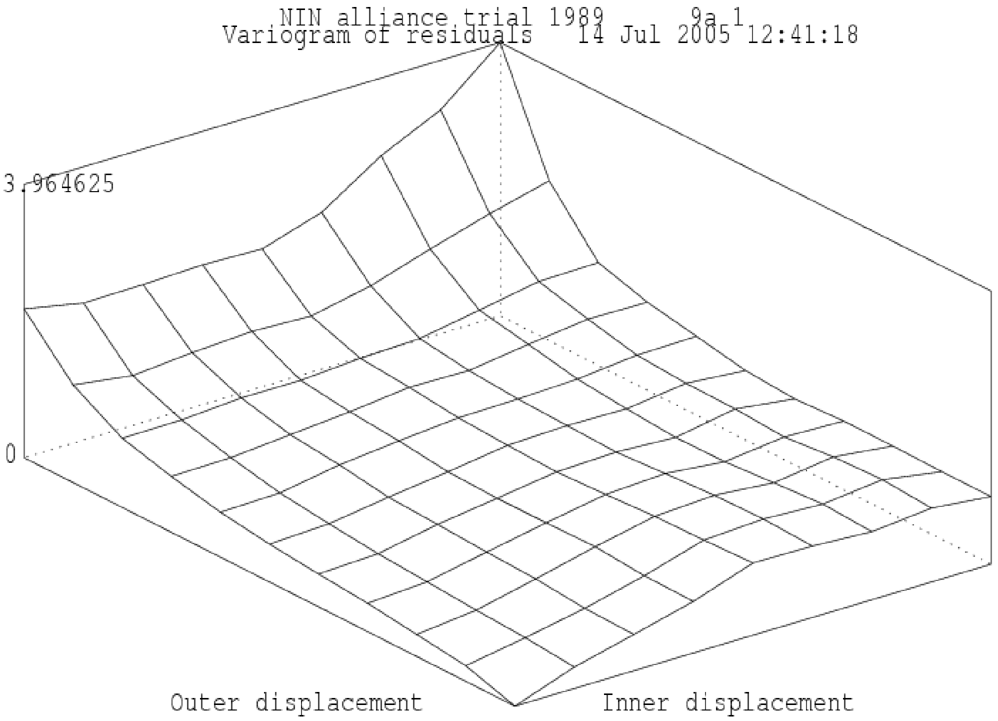

Figure 14.2: Variogram of residuals

Figures 14.2 to 14.5 show the graphics derived from the residuals when the `!DISPLAY 15` qualifier is specified and which are written to `.eps` files by running

```
ASReml -g22 nin89a.as
```

The graphs are a variogram of the residuals from the spatial analysis for site 1 (Figure 14.2), a plot of the residuals in field plan order (Figure 14.3), plots of the marginal means of the residuals (Figure 14.4) and a histogram of the residuals (Figure 14.5). The selection of which plots are displayed is controlled by the `!DISPLAY` qualifier (Table 5.4). By default, the variogram and field plan are displayed.

The sample variogram is a plot of the semi-variances of differences of residuals at particular distances. The (0,0) position is zero because the difference is identically zero. ASReml displays the plot for distances 0, 1, 2, ..., 8, 9-10, 11-14, 15-20, ...

The plot of residuals in field plan order (Figure 14.3) contains in its top and right margins a diamond showing the minimum, mean and maximum residual for that row or column. Note that a gap identifies where the missing values occur.

The plot of marginal means of residuals shows residuals for each row/column as well as the trend in their means.

Finally, we present a small example of the display produced when an XFA structure is fitted. The output from a small example with 9 environments and 2 factors is

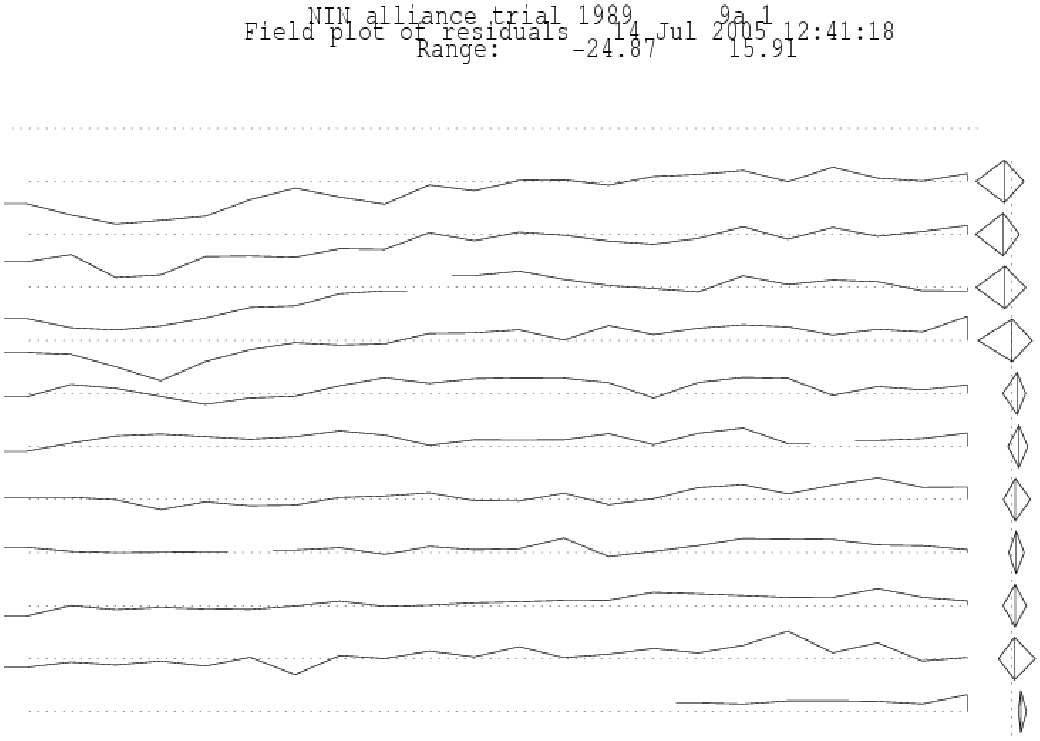

Figure 14.3: Plot of residuals in field plan order

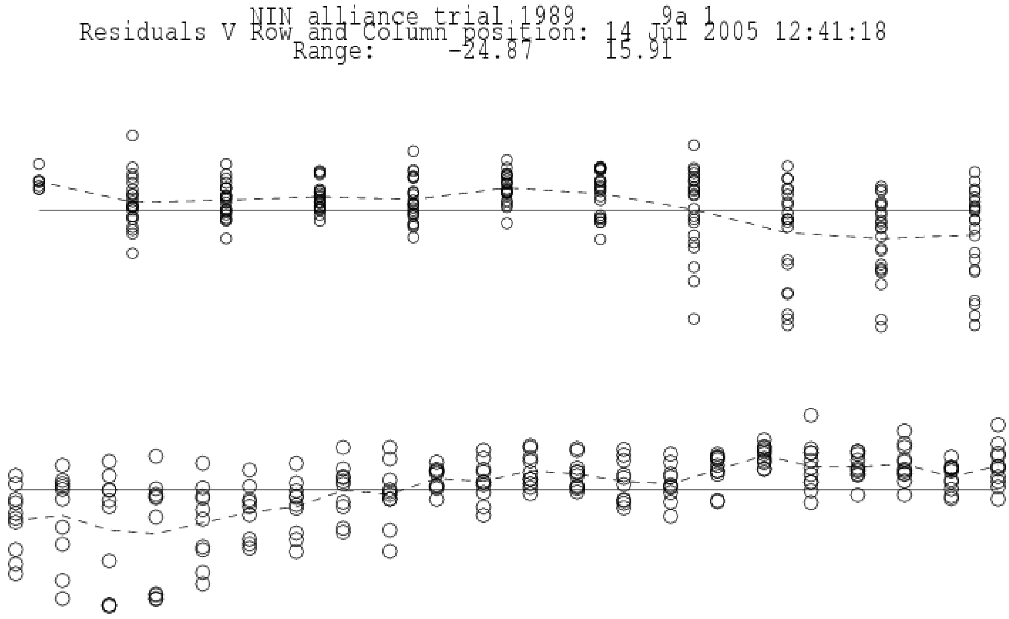

Figure 14.4: Plot of the marginal means of the residuals

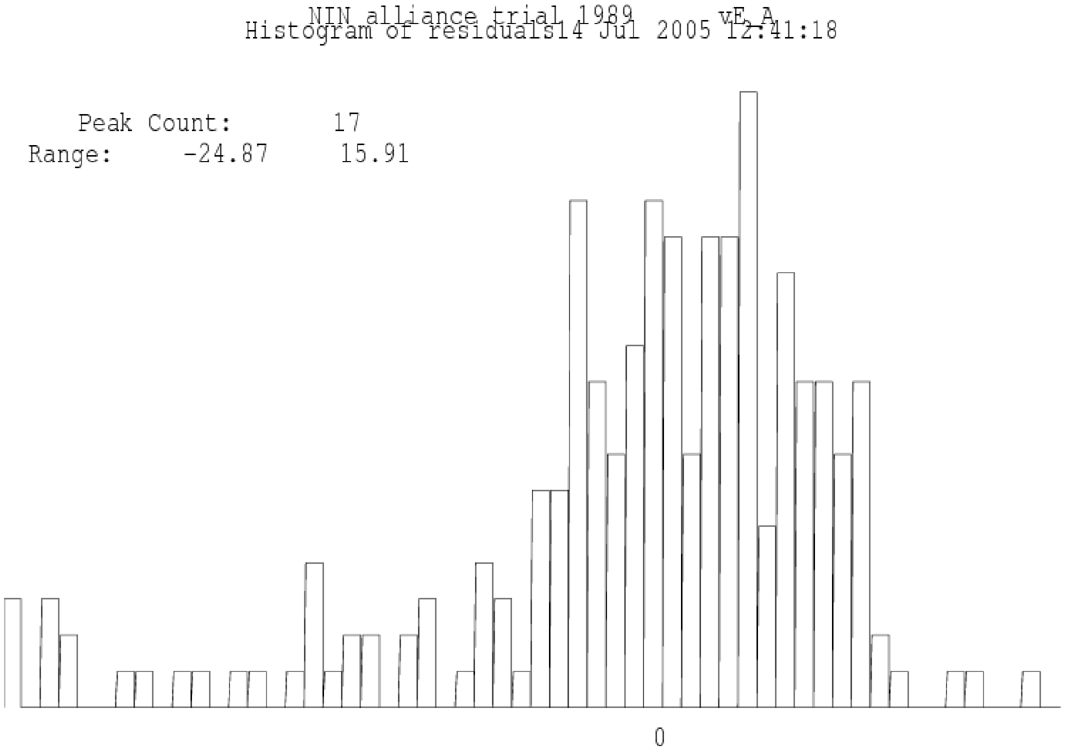

Figure 14.5: Histogram of residuals

```
DISPLAY of variance partitioning for XFA structure in xfa(Env,2).Geno
Lvl |----+----+----+----+----+----+----+----+----+----| TotalVar %expl PsiVar Loadings
  1 |                                       1         |   0.3339  79.7 0.0679 0.5147 0.0335
  2 |                                               1 2   0.1666 100.0 0.0000 0.4003 0.0797
  3 |                            1    2               |   0.2475  67.8 0.0798 0.3805 0.1514
  4 |                                            1    2   0.1475 100.0 0.0000 0.3625 0.1269
  5 |                                        1        2   0.4496 100.0 0.0000 0.6104 -0.278
  6 |                     1                           2   0.1210 100.0 0.0000 0.2287 0.2622
  7 |                    1     2                      |   0.4106  54.4 0.1872 0.4152 -0.226
  8 |    1                                            2   0.0901 100.0 0.0000 0.0922 0.2857
  9 |                           1                     2   0.1422 100.0 0.0000 0.2819 0.2506
  0 |----+----+----+----+----+----+----+----+-- Average   0.2343  89.1 0.0372 0.3651 0.0763
```

In the figure, 1 indicates the proportion of TotalVar explained by the first loading, 2 indicates the proportion explained by first and second (provided it plots right of 1. Consequently, the distance from 2 to the right margin represents PsiVar. %expl reports the percentage of TotalVar explained by all loadings. The last row contains column averages.

14.4.8 The .rsv file

The .rsv file contains the variance parameters from the most recent iteration of a model. The primary use of the .rsv file is to supply the values for the !CONTINUE qualifier (see Table 5.4) and the C command line option (see Table 11.1). It contains sufficient information to match terms so that it can be used when the variance model has been changed. This is nin89a.rsv.

```
 76 6 1711 121
```

```
# This .rsv file holds parameter values between runs of ASReml and
# is not normally modified by the User.  The current values of the
# the variance parameters are listed as a block on the following lines.
# They are then listed again with identifying information
# in a form that the user may edit.
   0.000000       0.000000       0.000000      1.0000000      0.6554798      0.4375045
RSTRUCTURE               1    2    3
 VARIANCE                1    1    0
   4, V, P,   1.00000000       0    0
 STRUCTURE              22    1    1
   5, R, P,   0.65547976       0    0
 STRUCTURE              11    1    1
   6, R, P,   0.43750453       0    0
```

14.4.9 The .tab file

The .tab file contains the simple variety means and cell frequencies. Below is a cut down version of nin89.tab.

```
  nin alliance trial                                       10 Sep 2002 04:20:15


                     Simple tabulation of yield


  variety
  LANCER                   28.56     4
  BRULE                    26.07     4
  REDLAND                  30.50     4
  CODY                     21.21     4
  ARAPAHOE                 29.44     4
  NE83404                  27.39     4
  NE83406                  24.28     4
  NE83407                  22.69     4
  CENTURA                  21.65     4
  SCOUT66                  27.52     4
  COLT                     27.00     4
⋮
  NE87615                  25.69     4
  NE87619                  31.26     4
  NE87627                  23.23     4
```

14.4.10 The .tsv file

The .tsv file contains the variance parameters as initialized for the most recent run in a form that is relatively easy to edit if the initial values need to be reset. The file is read when !TSV or !CONTINUE 2 is specified or if !CONTINUE is specified but no .rsv file exists. This is nin89a.tsv.

```
# This .tsv file is a mechanism for resetting initial parameter values
# by changing the values here and rerunning the job with !CONTINUE 2.
# You may not change values in the first 3 fields
#                    or RP fields where RP_GN is negative.

# Fields are:
# GN, Term, Type, PSpace, Initial_value, RP_GN, RP_scale.

   4, "Variance 1", V, P,   1.00000000    ,    4,    1
   5, "ar1(row).ar1(column);ar1v(row)_1", R, P,   0.10000000    ,    5,    1
   6, "ar1(row).ar1(column);ar1(column)_1", R, P,   0.10000000    ,    6,    1

# Valid values for Pspace are F, P, U and maybe Z.

# RP_GN and RP_scale define simple parameter relationships;
# RP_GN links related parameters by the first GN number;
# RP_scale must be 1.0 for the first parameter in the set and
# otherwise specifies the size relative to the first parameter.
# Multivalue RP_scale parameters may not be altered here.

# Notice that this file is overwritten if not being read.
```

14.4.11 The `.vrb` file

The `.vrb` file contains the estimates of the effects together with their approximate prediction variance matrix corresponding to the dense portion. It is only written if the `!VRB` qualifier is specified. The file is formatted for reading back for post processing. The number of equations in the dense portion can be increased (to a maximum of 800) using the `!DENSE` option (Table 5.5) but not to include random effects. The matrix is lower triangular row-wise in the order that the parameters are printed in the `.sln` file. It can be thought of as a partitioned lower triangular matrix,

$$\begin{bmatrix} \sigma^2 & . \\ \tilde{\boldsymbol{\beta}}_D & \sigma^2 C^{DD} \end{bmatrix}$$

where $\tilde{\boldsymbol{\beta}}_D$ is the dense portion of $\boldsymbol{\beta}$ and C^{DD} is the dense portion of C^{-1}. This is part of `nin89a.vrb`. Note that the first element is the estimated error variance, that is, 48.6802, see the variance component estimates in the `.asr` output.

```
   0.487026E+02   0.000000E+00   0.000000E+00   0.298409E+01   0.000000E+00
   0.807354E+01   0.470629E+01   0.000000E+00   0.456542E+01   0.886497E+01
  -0.315807E+00   0.000000E+00   0.409951E+01   0.476481E+01   0.876563E+01
   0.295379E+01   0.000000E+00   0.343250E+01   0.389543E+01   0.416076E+01
   0.743440E+01   0.163089E+01   0.000000E+00   0.377085E+01   0.428016E+01
   0.472451E+01   0.402633E+01   0.837086E+01   0.129027E+01   0.000000E+00
   0.329974E+01   0.347377E+01   0.357535E+01   0.316846E+01   0.412043E+01
   0.768099E+01   0.309076E+00   0.000000E+00   0.376552E+01   0.419706E+01
   0.395640E+01   0.383367E+01   0.458364E+01   0.378483E+01   0.984962E+01
   0.226400E+01   0.000000E+00   0.379190E+01   0.442373E+01   0.439411E+01
   0.402430E+01   0.440457E+01   0.362313E+01   0.502025E+01   0.901017E+01
   0.508505E+01   0.000000E+00   0.393519E+01   0.430418E+01   0.423685E+01
   0.428749E+01   0.417784E+01   0.363262E+01   0.444716E+01   0.527187E+01
   0.855044E+01   0.243553E+01   0.000000E+00   0.351279E+01   0.369901E+01
   0.383964E+01   0.330102E+01   0.361942E+01   0.352305E+01   0.359462E+01
   0.392014E+01   0.406704E+01   0.801337E+01   0.475798E+01   0.000000E+00
   0.370878E+01   0.418534E+01   0.452789E+01   0.408589E+01   0.446476E+01
   0.375742E+01   0.403945E+01   0.420473E+01   0.406937E+01   0.403049E+01
   0.857644E+01   0.606943E+00   0.000000E+00   0.428611E+01   0.506706E+01
   0.432088E+01   0.387484E+01   0.436861E+01   0.391305E+01   0.421110E+01
```

The first 5 rows of the lower triangular matrix are

```
48.7026
 0.0000  0.0000
 2.9841  0.0000  8.0735
 4.7063  0.0000  4.5654  8.8650
-0.3158  0.0000  4.0995  4.7648  8.7656
```

14.4.12 The `.vvp` file

The `.vvp` file contains the inverse of the average information matrix on the components scale. The file is formatted for reading back under the control of the `.pin` file described in Chapter 13. The matrix is lower triangular row-wise in the order the parameters are printed in the `.asr` file. This is `nin89a.vvp` with the parameter estimates in the order error variance, spatial row correlation, spatial column correlation.

```
 Variance of Variance components     3
   51.1980
  0.217689      0.317838E-02
  0.673382E-01 -0.201115E-02  0.649673E-02
```

14.5 ASReml output objects and where to find them

Table 14.2 presents a list of objects produced with each ASReml run and where to find them in the output files.

Table 14.2: ASReml output objects and where to find them

output object	found in	comment
Wald F statistics table	`.asr` file	This table contains Wald F statistics for each term in the *fixed* part of the model. These provide for an incremental or optionally a conditional test of significance (see Section 6.11).

14.5 ASReml output objects and where to find them

Table 14.2: Table of output objects and where to find them ASReml

output object	found in	comment
data summary	.asr file .ass file	includes the number of records read and retained for analysis, the minimum, mean, maximum, number of zeros, number of missing values per data field, factor/variate field distinction. An extended report of the data is written to the .ass file if the !SUM qualifier is specified. It includes cell counts for factors, histograms of variates and simple correlations among variates
eigen analysis	.res file	When ASReml reports a variance matrix to the .asr file, it also reports an eigen analysis of the matrix (eigen values and eigen vectors) to the .res file.
elapsed time	.asr file .asl file	this can be determined by comparing the start time with the finishing time. The execution times for parts of the Iteration process are written to the .asl file if the !DEBUG !LOGFILE command line qualifiers are invoked.
fixed and random effects	.sln file	if !BRIEF -1 is invoked, the effects that were included in the dense portion of the solution are also printed in the .asr file with their standard error, a t-statistic for testing that effect and a t-statistic for testing it against the preceding effect in that factor.
heritability	.pvc file	placed in the .pvc file when postprocessing with a .pin file
histogram of residuals	.res file	and graphics file
intermediate results	.asl file	given if the -DL command line option is used.
mean/variance relationship	.res file	for non-spatial analyses ASReml prints the slope of the regression of log(abs(residual)) against log(predicted value). This regression is expected to be near zero if the variance is independent of the mean. A power of the mean data transformation might be indicated otherwise. The suggested power is approximately (1-b) where b is the slope. A slope of 1 suggests a log transformation. This is indicative only and should not be blindly applied. Weighted analysis or identifying the cause of the heterogeneity should also be considered. This statistic is not reliable in genetic animal models or when units is included in the linear model because then the predicted value includes some of the residual.

14.5 ASReml output objects and where to find them

Table 14.2: Table of output objects and where to find them ASReml

output object	found in	comment
observed variance/ covariance matrix formed from BLUPs and residuals	.res file	for an interaction fitted as random effects, when the first [outer] dimension is smaller than the inner dimension less 10, ASReml prints an observed variance matrix calculated from the BLUPs. The observed correlations are printed in the upper triangle. Since this matrix is not well scaled as an estimate of the underlying variance component matrix, a rescaled version is also printed, scaled according to the fitted variance parameters. The primary purpose for this output is to provide reasonable starting values for fitting more complex variance structure. The correlations may also be of interest. After a multivariate analysis, a similar matrix is also provided, calculated from the residuals.
phenotypic variance	.pvc file	placed in the .pvc file when postprocessing with a .pin file
plot of residuals against field position	graphics file	
possible outliers	.res file	these are residuals that are more than 3.5 standard deviations in magnitude
predicted (fitted) values at the data points	.yht file	these in the are printed in the second column
predicted values	.pvs file	given if a predict statement is supplied in the .as file.
REML log-likelihood	.asr file	the REML log-likelihood is given for each iteration. The REML log-likelihood should have converged
residuals	.yht file	and in binary form in .dpr file; these are printed in column 3. Furthermore, for multivariate analyses the residuals will be in data order (traits within records). However, in a univariate analysis with missing values that are not fitted, there will be fewer residuals than data records - there will be no residual where the data was missing so this can make it difficult to line up the values unless you can manipulate them in another program (spreadsheet).
score	.asl file	given if the -DL command line option is used.
tables of means	.tab file .pvs file	simple averages of cross classified data are produced by the tabulate directive to the .tab file. Adjusted means predicted from the fitted model are written to the .pvs file by the predict directive.
variance of variance parameters	.vvp file	based on the inverse of the average information matrix
variance parameters	.asr file .res file	the values at each iteration are printed in the .res file. The final values are arranged in a table, printed with labels and converted if necessary to variances.
variogram	graphics file	

14.5 ASReml output objects and where to find them

Table 14.2: Table of output objects and where to find them ASReml

output object	found in	comment

15 Error messages

15.1 Introduction

Identifying the reason that ASReml does not produce the anticipated results can be a frustrating business. This chapter aims to assist you by discussing four kinds of errors. If ASReml does not run at all, it is a setup or licensing issue which is not discussed in this chapter. It is hoped that the new syntax for variance structure specification will reduce the incidence of coding errors.

Even when the job appears to run successfully, you should check that

- the records read/lines read/records used are correct,
- mean min max information is correct for each variable,
- the Loglikelihood has converged and the variance parameters are stable,
- the fixed effects have the expected degrees of freedom.

Coding errors can be classified as

- typing errors: these are difficult to resolve because we tend to read what we intended to type, rather than what we actually typed. Section 15.4 demonstrates the consequences of the common typographical errors that users make.
- wrong coding: this arises often from misunderstanding the guide or making assumptions arising from past experience which are not valid for ASReml. The best strategy here is to closely follow a worked example, or to build up to the required model. Sections 15.3 and 15.2 may help as well as reviewing all the relevant sections of this Guide. It may be as simple as adding or deleting a space, inserting a comma, changing case or adding one more qualifier.
- inappropriate model: the variance model you propose may not be suited to the data in which case ASReml may fail to produce a solution. You can verify the model is appropriate by closer examination of the structure of the data and by fitting simpler models.

- software problems: There are many options in ASReml and some combinations have not been tested. Some jobs are too big. When all else fails, describe your problem to the forum `http://www.vsni.co.uk/forum` or email `support@vsni.cu.uk`.

There are over 6000 one line diagnostic messages that ASReml may print in the `.asr` file. Hopefully, most are self explanatory, but it will always be helpful to recognise whether they relate to parsing the input file, or raise some other issue. See Section 15.5 for more information on these messages.

15.2 Common problems

Common problems in coding ASReml are as follows:

- a variable name has been misspelt; variable names are case sensitive,
- a model term has been misspelt; model term functions and reserved words (`mu`, `Trait`, `mv`, `units`) are case sensitive,
- the data file name is misspelt or the wrong path has been given - enclose the pathname in quotes (`'`) if it includes embedded blanks,
- a qualifier has been misspelt or is in the wrong place,
- failure to use commas appropriately in model definition lines,
- there is an error in the predict statement,
- model term `mv` not included in the model when there are missing values in the data and the model fitted assumes all data is present.
- there is an inconsistency between the variance header line and the structure definition lines presented (original syntax),
- there is an error in the R structure definition lines,
- there is an error in the G structure definition lines,
  - there is a factor name error,
  - there is a missing parameter,
  - there are too many/few initial values,

The most common problem in running ASReml is that a variable label is misspelt.

15.2 Common problems

The primary file to examine for diagnostic messages is the `.asr` file. When ASReml finds something atypical or inconsistent, it prints an diagnostic message. If it fails to successfully parse the input, it dumps the current information to the `.asr` file. Below is the output for a job that has been terminated due to an coding error. If a job has an error you should

- read the whole `.asr` file looking at all messages to see whether they identify the problem,

- focus particularly on any error message in the Fault: line and the text of the Last line read: (this line appears twice in the file to make it easier to find),

- check that all variables have been defined and are referenced with the correct case,

- some errors arise from conflicting information; the error may point to something that appears valid but is inconsistent with something earlier in the file,

- reduce to a simpler model and gradually build up to the desired analysis - this should help to identify the exact location of the problem.

If the problem is not resolved after these checks, you may need to email Customer Support at `support@asreml.co.uk`. Please send the `.as` file, (a sample of) the data, the `.asr` file and the `.asl` file produced by the debug options (`-dl`) running
`asreml -dl` *basename*`.as`

In this chapter we show some of the common coding problems. The code box on the right shows our familiar job modified to generate 8 faults. Following is the output from running this job.

```
NIN Alliance Trial 1989
variety *
id pid raw
repl *
nloc yield
lat long
row * column *
nin9.asd !slip 1
yield ~ mu variety
!R Repl
residual ar1(Row).ar1(Col)
predict varierty
```

```
 ASReml 4.1 [01 Apr 2014] NIN alliance trial 1989
   Build kt [21 Apr 2014]   64 bit  Windows x64
 23 Apr 2014 09:16:54.727     32 Mbyte  ninerr1
...
 Folder: C:\Users\Public\ASReml\Docs\Manex4\ERR
         There is no file called nin9.asd
         Variable names may not include "."
 Warning: Unrecognised qualifier at character    10 nin9.asd ! ... !SLIP 1         17
 Error: Failed to recognise a data file!
        Check spelling of filename and enclose the name in quotes.
 Fault: Error parsing yield ~ mu variety
  Last line read was:  yield ~ mu variety
  Currently defined structures, COLS and LEVELS
```

```
   1 variety                           1     2     0     0     0     0
   2 id                                1     1     0     0     0     0
   3 pid                               1     1     0     0     0     0
   4 raw                               1     1     0     0     0     0
   5 repl                              1     2     0     0     0     0
   6 nloc                              1     1     0     0     0     0
   7 yield                             1     1     0     0     0     0
   8 lat                               1     1     0     0     0     0
   9 long                              1     1     0     0     0     0
  10 row                               1     2     0     0     0     0
  11 column                            1     2     0     0     0     0
 ninerr1 C:\Users\Public\ASReml\Docs\Manex4\ERR
   11 factors defined [max5000].
    0 variance parameters [max2500].   2 special structures
  Last line read was:  yield ~ mu variety
 Finished: 23 Apr 2014 09:16:54.931   Error parsing yield ~ mu variety
```

ASReml happily reads down to the `nin9.asd` line. This name contains a '.' which is not permitted in a variable name so `nin9.asd` is expected to be a file name, but there is no such file in the working folder. The data file is actually `nin89.asd`.

15.3 Things to check in the `.asr` file

The information that ASReml dumps in the `.asr` file when an error is encountered is intended to give you some idea of the particular error:

- if there is no data summary, ASReml has failed before or while reading the model line,
- if ASReml has completed one iteration the problem is probably associated with starting values of the variance parameters or the logic of the model rather than the syntax *per se*.

15.4 An example

Briefly, the 8 coding errors in the example above, in the order they will be detected, are:

1. filename misspelt; there is no file `nin9.asd` in the working folder,

2. unrecognised qualifier (should be `!SKIP`),

3. 'Variety' has alphabetic level labels but not declared has such; `!A` required,

4. comma missing from first line of model; `!R Repl` is part of the model but not recognised as such,

5. misspelt variable label in linear model; `Repl` should be `repl`,

6. misspelt variable labels in `residual` model

7. the data has missing cells with respect to the declared residual structure,

8. misspelt variable label in predict; statement (`varierty` should be `variety`).

1. Data file not found

Running this job produces the `.asr` file in Section 15.1. The first problem is that ASReml cannot find a data file `nin9.asd` in the current working folder as indicated in the error message above the `Fault` line. Since `nin9.asd` contains a '.' which is not permitted in variable names, ASReml checks for a file of this name (in the working directory since no path is supplied). But ASReml did not find a file with this name. ASReml cannot tell whether the filename is misspelt or that an invalid variable name has been specified. In this case the data file was given as `nin9.asd` rather than `nin89.asd`. However, ASReml kept going and read the model line which it recognised because of the ∼ character. The message `Fault: Error parsing yield ∼ mu variety` does not mean that the error is in the model `yield ˜ mu variety` but that it recognised this as the model line and gave up because it had not encountered a valid data file line.

The message
`Warning: Unrecognised qualifier at character 10 nin9.asd !  ...  !SLIP 1`
simply indicates that the qualifier `!SLIP 1` has not been processed.

2. An unrecognised qualifier

After correcting the filename, we get the following (abbreviated) output. The problem is that `!SKIP 1`, which would cause ASReml to skip the first line of the data file, was mistyped as `!SLIP 1` which ASReml failed to recognise and ignored. But then it was unable to read the first line of the data file.

```
NIN Alliance Trial 1989
variety *
...
row * column *
nin89.asd !slip 1
yield ∼ mu variety
!R Repl
residual ar1(Row).ar1(Col)
predict varierty
```

```
...
Folder: C:\Users\Public\ASReml\Docs\Manex4\ERR
 QUALIFIERS: !SLIP 1
 Warning: Unrecognised qualifier at character  11 !SLIP 1
 Reading nin89.asd  FREE FORMAT skipping     0 lines

 Univariate analysis of yield
 Notice: Maybe you want !A !L qualifiers for this factor: variety
 Error at field  1 [variety] of record       1 [line       1]
 Since this is the first data record, you may need to skip some header lines
 (see !SKIP) or append the !A qualifier to the definition of factor variety
 Fault: Missing/faulty !SKIP or !A needed for variety
  Last line read was:  variety id pid raw rep nloc yield lat long row column
  Currently defined structures, COLS and LEVELS
   1 variety                          1     2     2     0     0     0
  ...
  10 row                              1     2     2     0     9     0
  11 column                            1     2     2     0    10     0
  12 mu                                0     1    -8     0    -1     0
 ninerr2 nin89.asd
  Model specification:  TERM LEVELS GAMMAS
 mu                                       0
 variety                                  0
   12 factors defined [max5000].
    0 variance parameters [max2500].   2 special structures
  Last line read was:  variety id pid raw rep nloc yield lat long row column
 Finished: 23 Apr 2014 09:17:01.765   Missing/faulty !SKIP or !A needed for variety
```

3. An incorrectly defined factor

After correcting `!slip 1` to `!skip 1`, we get the following (abbreviated) output. The problem is that `variety` is coded in the data file with alphabetic level names but ASReml is expecting integer level codes. Changing the `variety *` line to read `variety !A` resolves this problem.

```
NIN Alliance Trial 1989
variety *
...
row * column *
nin89.asd !skip 1
yield ∼ mu variety
!R Repl
residual ar1(Row).ar1(Col)
predict varierty
```

```
...
 Folder: C:\Users\Public\ASReml\Docs\Manex4\ERR
 QUALIFIERS: !SKIP 1
 Reading nin89.asd  FREE FORMAT skipping     1 lines

 Univariate analysis of yield
 Notice: Maybe you want !A !L qualifiers for this factor: LANCER
 Error at field   1 [LANCER] of record       1 [line       1]
 Since this is the first data record, you may need to skip some header lines
 (see !SKIP) or append the !A qualifier to the definition of factor variety
 Fault: Missing/faulty !SKIP or !A needed for variety
  Last line read was:  LANCER 1 1101 585 1 4 29.25 4.3 19.2 16 1
  Currently defined structures, COLS and LEVELS
   1 variety                           1     2     2     0     0     0
   ...
  11 column                            1     2     2     0    10     0
  12 mu                                0     1    -8     0    -1     0
 ninerr3 variety id pid raw rep nloc yield lat
  Model specification:  TERM LEVELS GAMMAS
 mu                             0
 variety                        0
   12 factors defined [max5000].
    0 variance parameters [max2500].   2 special structures
  Last line read was:  LANCER 1 1101 585 1 4 29.25 4.3 19.2 16 1
 Finished: 23 Apr 2014 09:17:05.540   Missing/faulty !SKIP or !A needed for variety
```

4. A missing comma

After correcting the definition of `variety`, we get the following (abbreviated) output. We have at least now read the data file as indicated by the data summary. You should always check the data summary to ensure that the correct number of records have been detected and the data values match the names appropriately.

```
NIN Alliance Trial 1989
variety !A
...
row * column *
nin89.asd !skip 1
yield ~ mu variety
!R Repl
residual ar1(Row).ar1(Col)
predict varierty
```

The problem is that `!R Repl` is meant to be part of the linear model, but it is on a separate line, and the first part of the model on the preceding line does not end with a COMMA to indicate that the model is incomplete. Appending a comma to the first model line resolves this problem.

```
...
 Folder: C:\Users\Public\ASReml\Docs\Manex4\ERR
 variety !A
 QUALIFIERS: !SKIP 1
 Reading nin89.asd  FREE FORMAT skipping     1 lines

 Univariate analysis of yield
 Summary of 224 records retained of 224 read

  Model term          Size #miss #zero   MinNon0    Mean      MaxNon0  StndDevn
   1 variety            56     0     0      1    28.5000      56
   2 id                        0     0  1.000      28.50      56.00      16.20
   3 pid                       0     0  1101.      2628.      4156.      1121.
   4 raw                       0     0  21.00      510.5      840.0      149.0
   5 repl                4     0     0      1     2.5000       4
   6 nloc                      0     0  4.000      4.000      4.000      0.000
   7 yield          Variate    0     0  1.050      25.53      42.00      7.450
   8 lat                       0     0  4.300      27.22      47.30      12.90
   9 long                      0     0  1.200      14.08      26.40      7.698
  10 row                22     0     0      1    11.7321      22
  11 column             11     0     0      1     6.3304      11
  12 mu                              1
 QUALIFIERS: !R Repl
 Fault: Error in variance header line: !R Repl
  Last line read was:  !R Repl 0 0 0 0
 ninerr4 variety id pid raw rep nloc yield lat
  Model specification:  TERM LEVELS GAMMAS
 variety                                56
 mu                                      1
   12 factors defined [max5000].
    0 variance parameters [max2500].   2 special structures
 Final parameter values [  2:   0]
  Last line read was:  !R Repl 0 0 0 0
 Finished: 23 Apr 2014 09:17:08.861   Error in variance header line: !R Repl
```

5. A misspelt factor name in linear model.

After correcting the definition of `variety`, we get the following (abbreviated) output. Now it has failed to parse the model line because the model term `Repl` was declared as `repl` and so is unrecognised. Changing `Repl` to `repl` (or vice versa) resolves this problem.

```
NIN Alliance Trial 1989
variety !A
id pid raw
repl *
...
row * column *
nin89.asd !skip 1
yield ~ mu variety ,
!R Repl
residual ar1(Row).ar1(Col)
predict varierty
```

```
...
 Folder: C:\Users\Public\ASReml\Docs\Manex4\ERR
 variety !A
 QUALIFIERS: !SKIP 1
 Reading nin89.asd  FREE FORMAT skipping     1 lines
 Model term "Repl" is not valid/recognised.
 Fault:  Error reading model terms
  Last line read was:     Repl
  Currently defined structures, COLS and LEVELS
   1 variety                          1     2     2     0     0     0
   2 id                               1     1     1     0     1     0
   3 pid                              1     1     1     0     2     0
   4 raw                              1     1     1     0     3     0
   5 repl                             1     2     2     0     4     0
...
  12 mu                               0     1    -8     0    -1     0
 ninerr5 variety id pid raw rep nloc yield lat
  Model specification:  TERM LEVELS GAMMAS
 mu                            0
 variety                       0
   12 factors defined [max5000].
    0 variance parameters [max2500].   2 special structures
  Last line read was:     Repl
 Finished: 23 Apr 2014 09:17:15.785    Error reading model terms
```

6. Misspelt factor name in RESIDUAL declaration

After correcting the spelling of `Repl`, we get the following (abbreviated) output. The problem here is essentially the same as error 5. The spatial residual model was declared using `Row` and `Col` but the relevant variables are in fact `row` and `column`. Note that, in this case `column` could be truncated to `col` in the model formulae as this does not cause any ambiguity but often it is clearer to use the full variable name.

```
NIN Alliance Trial 1989
variety !A
id pid raw
repl *
...
row * column *
nin89.asd !skip 1
yield ~ mu variety ,
!R repl
residual ar1(Row).ar1(Col)
predict varierty
```

```
...
 Summary of 224 records retained of 224 read

  Model term          Size #miss #zero   MinNon0    Mean      MaxNon0  StndDevn
   1 variety           56     0     0      1    28.5000         56
...
  10 row               22     0     0      1    11.7321         22
  11 column            11     0     0      1     6.3304         11
  12 mu                          1
 ar1(Row) in ar1(Row).ar1(Col) has size 0, parameters:   5   5
 ar1(Col) in ar1(Row).ar1(Col) has size 0, parameters:   6   6
  ar1(Row).ar1(Col)                [  4:  6] initialized.
 Error: There are 224 data records but RESIDUAL model implies 0 data records.

   Error: Unrecognised argument in ar1(Row)
   Error: Unrecognised argument in ar1(Col)
 Fault: RESIDUAL structure does not match records in data
  Last line read was:  Residual ar1(Row).ar1(Col)
 ninerr6 variety id pid raw rep nloc yield lat
  Model specification:  TERM LEVELS GAMMAS
 variety                                56
 mu                                       1
 repl                                     4     0.100 [  3]
SECTIONS        0        4        1
  STRUCT        0        1        1        5        1        1        1
   17 factors defined [max5000].
    6 variance parameters [max2500].   2 special structures
 Final parameter values [  3:   6]             0.10000E+00 1.00000     0.10000E+00
   0.10000E+00
  Last line read was:  Residual ar1(Row).ar1(Col)
 Finished: 23 Apr 2014 09:17:20.179   RESIDUAL structure does not match records in data
```

7. Missing plots in field layout

```
NIN Alliance Trial 1989
variety !A
id pid raw
repl *
...
row * column *
nin89.asd !skip 1
yield ∼ mu variety ,
!r repl
residual ar1(row).ar1(col)
predict varierty
```

The variables `row` and `column` define a 22 × 11 grid, that is 242 plots, but there are only 224 plots in the data. We could manually work out which are missing, and construct extra data lines to complete the grid, but ASReml will do this for us if we add the qualifiers
`!ROWFAC row !COLFAC column`
and add the model term `mv` to estimate missing values for the missing plots. So this problem is resolved by changing the model lines to read

```
nin89.asd !skip 1
!ROWFAC row !COLFAC column
yield ∼ mu variety mv !R repl
residual ar1(row).ar1(col)
```

This output also flags the 8th error which is the misspelling of `variety` in the predict line. That error does not stop the job running, but does mean the predicted means for variety will not be formed.

```
...
 QUALIFIERS: !SKIP 1
 Reading nin89.asd  FREE FORMAT skipping     1 lines
...
  12 mu                          1
 ar1(row) in ar1(row).ar1(col) has size 22, parameters:   5   5
 ar1(col) in ar1(row).ar1(col) has size 11, parameters:   6   6
  ar1(row).ar1(col)                [  4:  6] initialized.
 Forming      61 equations:  57 dense.
 Initial updates will be shrunk by factor    0.400

 Notice: Invalid argument, unrecognised qualifier or
         vector space exhausted at 'varierty   '

   Error: R structures do not match records in data.
   Error: Spatial Layout is not rectangular grid
 Fault: Variance structure does not match data
  Last line read was:  !STOP
 ninerr7 variety id pid raw rep nloc yield lat
  Model specification:  TERM LEVELS GAMMAS
 variety                      56
 mu                            1
 repl                          4     0.100 [  3]
SECTIONS     242       4       1
  STRUCT      22       1       1       5       1       1      10
              10       1       1       6       1       1      11
   15 factors defined [max5000].
    6 variance parameters [max2500].   2 special structures
 Final parameter values [  3:   6]             0.10000      1.0000     0.10000
```

```
   0.10000
  Last line read was:  !STOP
 Finished: 23 Apr 2014 09:17:23.354   Variance structure does not match data
```

8. A misspelt factor name in the predict statement

The final error in the job is that a factor name is misspelt in the predict statement. This is a non-fatal error. The `.asr` file contains the messages

```
 Notice: Invalid argument, unrecognised qualifier or
         vector space exhausted at 'voriety    '
 Warning: Extra lines on the end of the input file are ignored from
 predict varierty
```

The faulty statement is otherwise ignored by ASReml and no `.pvs` file is produced. To rectify this statement correct `varierty` to `variety`.

15.5 Information, Warning and Error messages

ASReml prints information, warning and error messages in the `.asr` file. The major information messages are in Table 15.1. A list of warning messages together with the likely meaning(s) is presented in Table 15.2. Other error messages with their probable cause(s) is presented in Table 15.3.

Not all messages are listed here. If not, identify whether the problem is syntactical (as in the previous section), whether it is a processing problem (the job starts to process but does not complete) or a reporting problem:

- for a syntax problem, note that the actual problem may be in an earlier line, and the current message is indicating an inconsistency with what ASReml has already read. Scan the output for other messages which might indicate the problem. If the problem is not evident, simplify the job until the simpler version runs and then build back to the required model. Remember that the model statement is parsed before the data file is read, but any following statements (e.g. `residual`, `predict`) are parsed after the data is read.

- processing errors are indicated if the `.asr` file contains lines like
  ```
  Forming 18211 equations:  42 dense.
  Initial updates will be shrunk by factor 0.316
  ```
  Simple things to try are increasing `!WORKSPACE` and simplifying the model.

- reporting problems are indicated if the LogL has converged or ASReml has completed the specified number of iterations.

Do not hesitate to seek help on the forum and to report problems to `support@vsni.co.uk`. Often a simple solution is available.

15.5 Information, Warning and Error messages

Table 15.1: Some information messages and comments

information message	comment
`Logl converged`	the REML log-likelihood last changed less than 0.002 * iteration number and variance parameter values appear stable.
`BLUP run done`	A full iteration has not been completed. See discussion of `!BLUP`.
`JOB ABORTED by USER`	See discussion of `ABORTASR.NOW`.
`Logl converged, parameters not converged`	the change in REML log-likelihood was small and convergence was assumed but the parameters are, in fact, still changing.
`Logl not converged`	the maximum number of iterations was reached before the REML log-likelihood converged. The user must decide whether to accept the results anyway, to restart with the `!CONTINUE` command line option (see Section 11.3 on job control), or to change the model and/or initial values before proceding. The sequence of estimates is reported in the `.res` file. It may be necessary to simplify the model and estimate the dominant components before estimating other terms if the LogL is oscilating.
`Warning: Only one iteration performed`	Parameter values are not at the REML solution.
`Parameters unchanged after one iteration.`	Parameters appear to be at the REML solution in that the parameter values are stable.

Messages beginning with the word `Notice:` are not generally listed here. They provide information the user should be aware of as it may affect the interpretation of results. They are not in themselves errors in that the syntax is valid, but they may reflect errors in the sense that the user may have intended something different.

Messages beginning with the word `Warning:` highlight information that the user should check. Again, it may reflect an error if the user has intended something different.

Messages beginning with the word `Error:` indicate that something is inconsistent as far as ASReml is concerned. It may be a coding error that the user can fix easily, or a processing error which will generally be harder to diagnose. Often, the error reported is a symptom of something else being wrong.

15.5 Information, Warning and Error messages

Table 15.2: List of warning messages and likely meaning(s)

warning message	likely meaning
`Notice: ASReml has merged design points closer than`	This is to reduce the number of knot points used in fitting a spline.
`Warning:` *e* `missing values generated by !^ transformation`	data values should be positive.
`Warning:` *i* `singularities in AI matrix`	usually means the variance model is overparameterized. Look up `!AISING`.
`Warning:` *m* `variance structures were modified`	the structures are probably at the boundary of the parameter space.
`Warning:` *n* `missing values were detected in the design`	either use `!MVINCLUDE` or delete the records.
`Warning:` *n* `negative weights`	it is better to avoid negative weights unless you can check ASReml is doing the correct thing with them.
`Warning:` *r* `records were read from multiple lines`	check the data summary has the correct number of records, and all variables have valid data values. If ASReml does not find sufficient values on a data line, it continues reading from the next line.
`WARNING` *term* `has more levels [ ## ] than expected [ ## ]:`	You have probably mis-specified the number of levels in the factor or omitted the `!I` qualifier (see Section 5.4 on data field definition syntax). ASReml corrects the number of levels.
`Warning:` *term* `in the predict !IGNORE list`	the term did not appear in the model.
`Warning:` *term* `in the predict !USE list`	the term did not appear in the model.
`Warning:` *term* `is ignored for prediction`	terms like `units` and `mv` cannot be included in prediction.
`Warning: Check if you need the !RECODE qualifier`	`!RECODE` may be needed when using a pedigree and reading data from a binary file that was not prepared with ASReml.
`Warning: Code B - fixed at a boundary (!GP)`	suggest drop the term and refit the model.
`Warning: Dropped records were not evenly distributed across`	`!MVREMOVE` has been used to delete records which have a missing value in design variables. This has resulted in multivariate data no longer having an $n \times t$ (n subjects with t traits each) structure. This will be a problem if the R structure model assumes $n \times t$ data structure.
`Warning: Eigen analysis check of US matrix skipped`	the matrix may be OK but ASReml has not checked it.
`WARNING: Extra lines on the end of the input file ...:`	this indicates that there are some lines on the end of the .as file that were not used. The first ”extra” line is displayed. This is only a problem if you intended ASReml to read these lines.

15.5 Information, Warning and Error messages

Table 15.2: List of warning messages and likely meaning(s)

warning message	likely meaning
`Warning: Failed to find header blocks to skip.`	The `!RSKIP` qualifier requested skipping header blocks which were not present.
`Warning: Fewer levels found in` *term*	ASReml increases to the correct value.
`Warning: FIELD DEFINITION lines should be INDENTED`	indent them to avert this message.
`Warning: Fixed levels for factor`	user nominated more levels than are permitted.
`Warning: Initial gamma value is zero`	constraint parameter is probably wrongly assigned.
`Warning: Invalid argument.`	fix the argument.
`Warning: It is usual to include Trait in the ... model`	The model term `Trait` was not present in the multivariate analysis model.
`Warning: LogL Converged; Parameters Not Converged`	you may need more iterations.
`Warning: LogL not converged`	restart to do more iterations (see `!CONTINUE`).
`Notice: LogL values are reported relative to a base of`	The computed LogL value is occasionally very large in magnitude, but our interest is in relative changes. Reporting relative to an offset ensures that differences at the units level are apparent.
`Warning: Missing cells in table`	missing cells are normally not reported.
`Warning: More levels found in` *term*	consider setting levels correctly.
`Warning: PREDICT LINE IGNORED - TOO MANY`	the limit is 100 PREDICT statements.
`Warning: PREDICT statement is being ignored`	because it contains errors.
`Warning: Second occurrence of` *term* `dropped`	if you really want to fit this term twice, create a copy with another name.
`Warning: Spatial mapping information for side`	gives details so you can check ASReml is doing what you intend.
`Warning: Standard errors`	that is, these standard errors are approximate.
`Warning: SYNTAX CHANGE:` *text* `may be invalid`	use the correct syntax.
`Warning: The !A qualifier ignored when reading BINARY data`	the `!A` fields will be treated as factors but are coded as they appear in the binary file.
`Warning: The !SPLINE qualifier has been redefined.`	use correct syntax.

15.5 Information, Warning and Error messages

Table 15.2: List of warning messages and likely meaning(s)

warning message	likely meaning
`Warning: The !X !Y !G qualifiers are ignored. There is no data to plot`	revise the qualifier arguments.
`Warning: Warning: The default action with missing values in multivariate data`	The issue is to match the declared R structure to the physical data. Dropping observations which are missing will often usually destroy the pattern. Estimating missing values allows the pattern to be retained.
`Warning: The estimation was ABORTED`	Do not accept the estimates printed.
`Warning: The FOWN test of ... is not calculated ...`	The FOWN test requested is not calculated because it results in different numbers of degrees of freedom to that obtained for the incremental tests for the terms in the model as fitted; the FOWN calculations are based on the reduced design matrix formed for the incremental model. ASReml performs the standard conditional test instead. The user must reorder (swap?) the terms in the model specification and rerun the job to perform the requested FOWN test.
`Warning: The labels for predictions are erroneous`	the labels for predicted terms are probably out of kilter. Try a simpler predict statement. If the problem persists, send for help.
`Warning: This US structure is not positive definite`	check the initial values.
`Warning: Unrecognised qualifier at character`	the qualifier either is misspelt or is in the wrong place.
`Warning: US matrix was not positive definite: MODIFIED`	the initial values were modified by a 'bending' process.
`Warning: User specified spline points`	the points have been rescaled to suit the data values.
`Warning: Variance parameters were modified by BENDing`	ASReml may not have converged to the best estimate.
`Warning: Likelihood decreased. Check gammas and singularities.:`	a common reason is that some constraints have restricted the gammas. Add the `!GU` qualifier to any factor definition whose gamma value is approaching zero (or the correlation is approaching (-)1. Alternatively, more singularities may have been detected. You should identify where the singularities are expected and modify the data so that they are omitted or consistently detected. One possibility is to centre and scale covariates involved in interactions so that their standard deviation is close to 1.

Table 15.3: Alphabetical list of error messages and probable cause(s)/remedies

error message	probable cause/remedy

`!COLFAC confusion`
`!ROWFAC confusion`
	!COLFAC/!ROWFAC arguments contradict RESIDUAL statement order. If the variables have the correct names, reverse the order.

`!PRINT: Cannot open output file`	Check filename.
`!SUBSECTION not permitted ...`	This variance structure qualifier is only permitted in single section RESIDUAL structures.
`AINV/GIV matrix undefined or wrong size`	Check the size of the factor associated with the AINV/GIV structure.
`ALNORM Error`	ALNORM calculates the Normal Integral
`Apparent error in pedigree relationships`	ASReml failed to !SORT the pedigree.
`ASReml command file is EMPTY:`	The job file should be in ASCII format.
`ASReml failed in ...`	Try running the job with increased workspace, or using a simpler model. Otherwise send the job to VSN (mailto:`support@asreml.co.uk`) for investigation.
`at() string too long`	ASReml failed to expand the at() model term string. Break it into several parts on separate lines.
`Badly formed model term.`	ASReml failed to parse the term. Revise and simplify.
`CALC ??  reference to large.`	An argument in the CALC statement is not valid.
`Check IDV structure.`	ASReml is using IDV variance structure but wonders whether that is what you intended.

`Context of read error`
`Data Error:  At record ...`
	ASReml found a alphacharacters when it was expecting numeric data. Either the variable should be declared alphanumeric, or we have miscounted items on the line. Use !CSV if there are TAB or COMMA delimited blank lines.

`Continue from .rsv file`	Try running without the `!CONTINUE` qualifier.
`Convergence failed`	the program did not proceed to convergence because the REML log-likelihood was fluctuating wildly. One possible reason is that some singular terms in the model are not being detected consistently. Otherwise, the updated G structures are not positive definite. There are some things to try: – define `US` structures as positive definite by using `!GP`, – supply better starting values, – fix parameters that you are confident of while getting better estimates for others (that is, fix variances when estimating covariances), – fit a simpler model, – reorganise the model to reduce covariance terms (for example, use `CORUH` instead of `US`.)

15.5 Information, Warning and Error messages

Table 15.3: Alphabetical list of error messages and probable cause(s)/remedies

error message	probable cause/remedy
`Correlation structure is not positive definite`	It is best to start with a positive definite correlation structure. Maybe use a structured correlation matrix.
`Data does not have # sections.`	The data does not match the RESIDUAL specification.
`Define structure for ...`	A variance structure should be specified for this term.
`Error: The indicated number of input fields exceeds the limit.`	The reported limit is hardcoded. The number of variables to be read must be reduced.
`Error in !CONTRAST label factor values`	The error could be in the variable(factor) name or in the number of values or the list of values.
`Error in !GROUP label factor values`	The list of values does not agree with the factor definition.
`Error in !SUBSET label factor values`	The error could be in the variable(factor) name or in the number of values or the list of values.
`Error in extended !ASSIGN`	The `!<` `!>` qualifiers allow an assign string to be defined over several lines. Maybe the string is too long.
`Error in R structure: model checks`	the error model is not correctly specified.
`Error opening file`	the file did not exist or was of the wrong file type (binary = unformatted, sequential).
`Error in list ...`	The PREDICT statement cannot be parsed.
`Error in PREDICT`	ASReml failed to form the PREDICT design matrix.
`Error in variance header line.`	This usually indicates the model has not been properly parsed and part is misinterpreted as a variance header line (old syntax where the residual statement was expected. When the model statement is written over several lines, incomplete lines must end with a PLUS or COMMA character.
`Error in Variance Parameter Constraint`	Check old syntax variance structure specification.
`Error opening file.`	Check the filename is correct and that the file is not open in another process.
`Error order`	ASReml has failed to determine an order for solving the mixed model equations. See `!EQORDER` for some discussion. Try increasing `!WORKSPACE`.
`Error parsing`	This error comes from the main read routine! or from the variable definition parsing routine.
`Error reading` *something*	There are several messages of this form where *something* is what ASReml is attempting to read. Either there is an error telling ASReml to read *something* when it does not need to, or there is an error in the way *something* is specified.

Table 15.3: Alphabetical list of error messages and probable cause(s)/remedies

error message	probable cause/remedy
`Error reading the data:`	the data file could not be interpreted: alphanumeric fields need the `!A` qualifier.
`Error reading the DATA FILENAME line`	data file name may be wrong
`Error reading the model factor list`	the model specification line is in error: a variable is probably misnamed.
`Error: Ran out of space to code records to sort them!'/`	Declare the levels in the !ROWFACTOR, !COLUMNFACTOR and !SECTION variables more accurately.
`Error setting constraints (!VCC) on variance components`	The `!VCC` constraints are specified last of all and require knowing the position of each parameter in the parameter vector.
`Error setting dependent variable`	the specified dependent variable name is not recognised.
`Error setting MBF design matrix: !MBF mbf(x,k) filename`	It is likely that the covariate values do not match the values supplied in the file. The values in the file should be in sorted order.
`Error sorting X,Y values`	`!ROWFAC` and `!COLFAC` and `!SECTION` as well as factors defining a `residual` structure must uniquely define grid points in the spatial array.
`Error structures are wrong size:`	the declared size of the error structures does not match the actual number of data records.
`Error when reading knot point values`	There is some problem on the `!SPLINE` line. It could be a wrong variable name or the wrong number of knot points. Knot points should be in increasing order.
`Failed forming R/G scores...?`	Try increasing workspace.
`Failed ordering Level labels`	The problem may be due to the use of the `!SORT` qualifier in the data definition section.
`Failed to find ...`	The PREDICT statement seems in error: the named factor is not present in the model.
`Failed to open !INCLUDE`	An !INCLUDE file could not be opened.

`Failed to parse R/G structure line`
`Failed to read R/G structure line`
	May be an unrecognised factor/model-term name or variance structure name or wrong count of initial values, possible on an earlier line. May be insufficient lines in the job.

`Failed to process MYOWNGDG files`	Check your MYOWNGDG program and the `.gdg` file.

`Failed when sorting pedigree ...`
`Failed when processing pedigree file ...`
	Maybe increase !WORKSPACE. Messages may identify a problem with the pedigree.

15.5 Information, Warning and Error messages

Table 15.3: Alphabetical list of error messages and probable cause(s)/remedies

error message	probable cause/remedy
`Failed while ordering equations.`	This indicates the job needs more memory than was allocated or is available. Try increasing the workspace or simplifying the model.
`FORMAT error reading ...`	Likely causes are – bad syntax or invalid characters in the variable names; variable names must not include any of these symbols; `!|-+(:#$` and ., – the data file name is misspelt, – there are too many variables declared or there is no valid *value* supplied with an arithmetic transformation option.
`G-structure header:  Factor order:`	there is a problem reading G structure header line.An earlier error (for example insufficient initial values) may mean the actual line read is not actually a G header line at all. A G header line must contain the name of a term in the linear model spelt exactly as it appears in the model.
`G structure:  ORDER 0 MODEL GAMMAS:`	a G structure line cannot be interpreted.
`G structure size does not match`	The size of the structure defined does not agree with the model term that it is associated with.
`Getting Pedigree:`	an error occurred processing the pedigree. The pedigree file must be ascii, free format with `ANIMAL`, `SIRE` and `DAM` as the first three fields.
`GLM Bounds failure`	ASReml failed to calculate the GLM working variables or weights. Check the data.
`Increase declared levels for factor ...`	Either the field has alphanumeric values but has not been declared using the `!A` qualifier, or there is not enough space to hold the levels of the factor. To 'increase the levels', insert the expected number of levels after the `!A` or `!I` qualifier in the field definition.
`Increase workspace ...`	Use `!WORKSPACE` *s* to increase the workspace available to ASReml. If the data set is not extremely big, check the data summary.
`Insufficient data read from file`	Maybe the response variable is all missing.
`Insufficient points for :`	there must be at least 3 distinct data values for a spline term
`Insufficient workspace.`	If ASReml has not obtained the maximum available workspace, then use `!WORKSPACE` to increase it. The problem could be with the way the model is specified. Try fitting a simpler model or using a reduced data set to discover where the workspace is being used.
`invalid analysis trait number`	The response variable nominated by the `!YVAR` command line qualifier is not in the data.

15.5 Information, Warning and Error messages

Table 15.3: Alphabetical list of error messages and probable cause(s)/remedies

error message	probable cause/remedy

`Invalid binary data`
`Invalid Binomial Variable`
	The data values are out of the expected range for binary/binomial data.

`Invalid definition of factor ...`	there is a problem with forming one of the *generated* factors. The most probable cause is that an interaction cannot be formed.
`Invalid error structure for Multivariate Analysis`	You must either use the `US` error structure or use the `!ASUV` qualifier (and maybe include `mv` in the model).
`Invalid factor in model:`	a term in the *model specification* is not among the terms that have been defined. Check the spelling.
`Invalid model factor ...  :`	there is a problem with the named variable.
`Invalid SOURCE in R structure definition`	The second field in the R structure line does not refer to a variate in the data.
`Invalid weight/filter column number:`	the weight and filter columns must be data fields. Check the data summary.
`Iteration aborted because of singularities`	See the discussion of `!AISINGULARITIES`.
`Iteration failed`	Maybe increase workspace or restructure/simplify the model.
`Matern:  ...`	Numerical problems calculating the Matérn function. If rescaling the X, Y cordinates so that the step size is closer to 1.0 does not resolve the issue, try `AEXP` instead.
`Maximum number of special structures exceeded`	special structures are weights, the Ainverse and GIV structures. The limit is 98 and so no more than 96 GIV structures can be defined.
`Maximum number of variance parameters exceeded`	The limit is 1500. It may be possible to restructure the job so the limit is not exceeded, assuming that the actual number of parameters to be estimated is less.
`Missing/faulty !SKIP or !A needed for ...`	ASReml failed to read the first data record. Maybe it is a heading line which should be skipped by using the `!SKIP` qualifier, or maybe the field is an alphanumeric field but has not been declared so with the `!A` qualifier.
`Missing values in design variables/factors`	You need to identify which design terms contain missing values and decide whether to delete the records containing the missing values in these variables or, if it is reasonable, to treat the missing values as zero by using `!MVINCLUDE`.
`Missing Value Miscount forming design`	More missing values in the response were found than expected.
`Missing values not allowed here:`	missing observations have been dropped so that direct product R structure does not match the multivariate data structure.
`Multiple trait mapping problem`	Maybe a trait name is repeated.

15.5 Information, Warning and Error messages

Table 15.3: Alphabetical list of error messages and probable cause(s)/remedies

error message	probable cause/remedy
`Negative Sum of Squares:`	This is typically caused by negative variance parameters; try changing the starting values or using the `!STEP` option. If the problem occurs after several iterations it is likely that the variance components are very small. Try simplifying the model. In multivariate analyses it arises if the error variance is (becomes) negative definite. Try specifying `!GP` on the structure line for the error variance.
`NFACT out of range:`	too many terms are being defined.
`No .giv file for`	Fix the argument to giv().
`No residual variation:`	after fitting the model, the residual variation is essentially zero, that is, the model fully explains the data. If this is intended, use the `!BLUP 1` qualifier so that you can see the estimates. Otherwise check that the dependent values are what you intend and then identify which variables explain it. Again, the `!BLUP 1` qualifier might help.
`Out of ...`	A program limit has been breached. Try simplifying the model.
`Out of memory ...`	use `!WORKSPACE` qualifier to increase the workspace allocation. It may be possible to revise the models to increase sparsity.
`Out of memory: forming design:`	factors are probably not declared properly. Check the number of levels. Possibly use the `!WORKSPACE` qualifier.
`Overflow forming !PRESENT table`	The predict table appears to be too big. Try increasng WORKSPACE, or predicting in parts.
`Overflow structure table:`	occurs when space allocated for the structure table is exceeded. There is room for three structures for each model term for which G structures are explicitly declared. The error might occur when ASReml needs to construct rows of the table for structured terms when the user has not formally declared the structures. Increasing g on the variance header line for the number of *G structures* (see ASReml User Guide: Structural Specification) will increase the space allocated for the table. You will need to add extra explicit declarations also.
`Pedigree coding errors:`	check the pedigree file and see any messages in the output. Check that identifiers and pedigrees are in chronological order.
`Pedigree factor has wrong size:`	the A-inverse factors are not the same size as the A-inverse. Delete the ainverse.bin file and rerun the job.
`Pedigree too big! or in error`	Typically this arises when there is a problem processing the pedigree file.
`POWER model setup error`	Check the details for the distance based variance structure.
`POWER Model: Unique points disagree with size`	Check the distances specified for the distance based variance structure.
`PROGRAM failed in ...`	Try increasing workspace. Otherwise send problem to VSN.

15.5 Information, Warning and Error messages

Table 15.3: Alphabetical list of error messages and probable cause(s)/remedies

error message	probable cause/remedy
`PROGRAMMING error:`	indicates ASReml has failed deep in its core. It is likely to be an interaction between the data and the variance model being fitted. Try increasing the memory, simplifying the model and changing starting values for the gammas. If this fails send the problem to the VSN (mailto:`support@asreml.co.uk`) for investigation.
`reading !SELF option`	Check the argument.
`Reading distances for POWER structure`	POWER structures are the spatial variance models which require a list of distances. Distances should be in increasing order. If the distances are not obtained from variables, the 'SORT' field is zero and the distances are presented after all the R and G structures are defined.
`Reading factor names:`	something is wrong in the terms definitions. It could also be that the data file is misnamed.
`reading Overdispersion factor`	Check the argument.
`READING OWN structures ...`	There is probably a problem with the output from MYOWNGDG. Check the files, including the time stamps to check the `.gdg` file is being formed properly.
`Reading the data:`	if you read less data than you expect, there are two likely explanations. First, the data file has less fields than implied by the data structure definitions (you will probably read half the expected number). Second, there is an alphanumeric field where a numeric field is expected.
`Reading Update step size:`	check the `!STEP` qualifier argument.
`Residual Variance is Zero:`	either all data is deleted or the model fully fits the data.

`R header SECTIONS DIMNS GSTRUCT`
`R structure header SITE DIM GSTRUCT`
`Variance header:  SEC DIM GSTRUCT`
	error with the variance header line. Often, some other error has meant that the wrong line is being interpreted as the variance header line. Commonly, the model is written over several lines but the incomplete lines do not all end with a comma.

`R structure error ORDER SORTCOL MODEL GAMMAS:`	an error reading the error model.
`R structures are larger than number of records`	Maybe you need to include `mv` in the model to stop ASReml discarding records with missing values in the response variable.

`REQUIRE !ASUV qualifier for this R structure`
`REQUIRE I x E R structure`
	Without the `ASUV` qualifier, the multivariate error variance MUST be specified as `US`.

`Scratch:`	Apparently ASReml could not open a scratch file to hold the transformed data. On unix, check the temp directory `//tmp` for old large scratch files.

Table 15.3: Alphabetical list of error messages and probable cause(s)/remedies

error message	probable cause/remedy
`Segmentation fault:`	this is a Unix memory error. It typically occurs when a memory address is outside the job memory. The first thing to try is to increase the memory workspace using the `!WORKSPACE` (see Section 11.3 on memory) command line option. Otherwise you may need to send your data and the .as files to Customer Support for debugging.

`Singularity appeared in AI matrix`
`Singularity in Average Information Matrix`
	See the discussion on `!AISINGULARITIES`

`SINGULARITY IN ...`	Problem performing the 'Regression Screen'

`Sorting data by !Section !Row ...`
`Sorting the data into field order`
	the field order coding in the spatial error model does not generate a complete grid with one observation in each cell; missing values may be deleted: they should be fitted. Also may be due to incorrect specification of number of rows or columns.

`STOP SCRATCH FILE DATA STORAGE ERROR:`	ASReml attempts to hold the data on a scratch file. Check that the disk partition where the scratch files might be written is not too full; use the `!NOSCRATCH` qualifier to avoid these scratch files.
`Structure/ Factor mismatch:`	the declared size of a variance structure does not match the size of the model term that it is associated with.
`Too many alphanumeric factor level labels:`	if the factor level labels are actually all integers, use the `!I` option instead. Otherwise, you will have to convert a factor with alphanumeric labels to numeric sequential codes external to ASReml so that an `!A` option can be avoided.
`Too many factors with !A or !I; max 100`	The data file may need to be rewritten with some factors recoded as sequential integers.
`Too many [max 20] dependent variables`	This is an internal limit. Reduce the number of response variables. Response variables may be grouped using the `!G` factor definition qualifier so that more than 20 actual variables can be analysed.
`Unable to invert R or G [US?] matrix:`	this message occurs when there is an error forming the inverse of a variance structure. The probable cause is a non positive definite (initial) variance structure (`US`, `CHOL` and `ANTE` models). It may also occur if an *identity by unstructured* (`ID`⊗`US`) error variance model is not specified in a multivariate analysis (including `!ASMV`), see Chapter 8. If the failure is on the first iteration, the problem is with the starting values. If on a subsequent iteration, the updates have caused the problem. You can specify `!GP` to force the matrix positive definite, and try reducing the updates by using the `!STEP` qualifier. Otherwise, you could try fitting an alternative parameterisation.
`Unable to invert R or G [CORR?] matrix:`	generally refers to a problem setting up the mixed model equations. Most commonly, it is caused by a non positive definite matrix.

Table 15.3: Alphabetical list of error messages and probable cause(s)/remedies

error message	probable cause/remedy
`Variance structure is not positive definite`	Use better initial values or a structured variance matricx that is positive definite.

`XFA model not permitted in R structures`
`XFA may not be used as an R structure`
	You may use `FA` or `FACV`. The R structure must be positive definite.

16 Examples

16.1 Introduction

In this chapter we present the analysis of a variety of examples. The primary aim is to illustrate the capabilities of ASReml in the context of analysing real data sets. We also discuss the output produced by ASReml and indicate when problems may occur. Statistical concepts and issues are discussed as necessary but we stress that the analyses are illustrative, not prescriptive.

16.2 Split plot design - Oats

The first example involves 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 each split into four sub-plots. The three varieties were randomly allocated to the three whole-plots while the four levels of nitrogen application were randomly assigned to the four sub-plots within each whole-plot. The data is presented in Table 16.1.

Table 16.1: A split-plot field trial of oat varieties and nitrogen application

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

A standard analysis of these data recognises the two basic elements inherent in the experiment. These are firstly the stratification of the experiment units, that is the blocks, whole-plots and sub-plots, and secondly, the treatment structure that is superimposed on the experimental material. The latter is of prime interest, in the presence of stratification. Thus the aim of the analysis is to examine the importance of the treatment effects while accounting for the stratification and restricted randomisation of the treatments to the experimental units. The ASReml input file is presented below.

```
split plot example
 blocks   6     # Coded 1...6 in first data field of oats.asd
 nitrogen !A 4  # Coded alphabetically
 subplots *     # Coded 1...4
 variety  !A 3  # Coded alphabetically
 wplots   *     # Coded 1...3
 yield
oats.asd !SKIP 2

yield ~ mu variety nitrogen variety.nitrogen !r idv(blocks) idv(blocks.wplots)
residual idv(units)
predict nitrogen   # Print table of predicted nitrogen means
predict variety
predict variety nitrogen !SED
```

The data fields were `blocks`, `wplots`, `subplots`, `variety`, `nitrogen` and `yield`. The first five variables are factors that describe the stratification or experiment design and treatments. The standard split plot analysis is achieved by fitting the model terms `blocks` and `blocks.wplots` as random effects. The `blocks.wplots.subplots` term is not listed in the model because this interaction corresponds to the experimental units and is automatically included as the residual term. The fixed effects include the main effects of both `variety` and `nitrogen` and their interaction. The tables of predicted means and associated standard errors of differences (SEDs) have been requested. These are reported in the `.pvs` file. Abbreviated output is shown below.

```
           - - - Results from analysis of yield - - -
 Akaike Information Criterion      424.76 (assuming 3 parameters).
 Bayesian Information Criterion    431.04

          Approximate stratum variance decomposition
 Stratum     Degrees-Freedom   Variance      Component Coefficients
 idv(blocks)            5.00    3175.06        12.0      4.0     1.0
 idv(blocks.wplots     10.00    601.331         0.0      4.0     1.0
 Residual Variance     45.00    177.083         0.0      0.0     1.0

 Model_Term                             Gamma         Sigma   Sigma/SE   % C
 blocks                  IDV_V     6   1.21116      214.477       1.27   0 P
 blocks.wplots           IDV_V    18  0.598937      106.062       1.56   0 P
 idv(units)                       72 effects
 Residual                SCA_V    72  1.000000      177.083       4.74   0 P
```

```
                                         Wald F statistics
    Source of Variation                  NumDF     DenDF    F-inc             P-inc
  7 mu                                       1       5.0   245.14             <.001
  4 variety                                  2      10.0     1.49             0.272
  2 nitrogen                                 3      45.0    37.69             <.001
  8 variety.nitrogen                         6      45.0     0.30             0.932
```

For simple variance component models such as the above, the default parameterisation for the variance component parameters is as the ratio to the residual variance. Thus ASReml prints the variance component ratio and variance compo for each term in the random model in the columns labelled `Gamma` and `Component` respectively.

A table of Wald F statistics is printed below this summary. The usual decomposition has three strata, with treatment effects separating into different strata as a consequence of the balanced design and the allocation of `variety` to whole-plots. In this balanced case, it is straightforward to derive the ANOVA estimates of the stratum variances from the REML estimates of the variance components. That is

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

The default output for testing fixed effects used by ASReml is a table of so-called incremental Wald F statistics. These Wald F statistics are described in Section 6.11. They are simply the Wald test statistics divided by the number of estimable effects for that term. In this example there are four terms included in the summary. The overall mean (denoted by `mu`) 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, defined by the model which includes those terms appearing above the current term given the variance parameters. For example, the test of `nitrogen` is calculated from the change in sums of squares for the two models `mu variety nitrogen` and `mu variety`. No refitting occurs, that is the variance parameters are held constant at the REML estimates obtained from the currently specified fixed model.

The incremental Wald statistics have an asymptotic χ^2 distribution, with degrees of freedom (df) given by the number of estimable effects (the number in the `DF` column). In this example, the incremental Wald F statistics are numerically the same as the ANOVA F statistics, and ASReml has calculated the appropriate denominator df for testing fixed effects. This is a simple problem for balanced designs, such as the split plot design, but it is not straightforward to determine the relevant denominator df in unbalanced designs, such as the rat data set described in the next section.

Tables of predicted means are presented for the nitrogen, variety, and variety by nitrogen tables in the `.pvs` file. The qualifier `!SED` has been used on the third predict statement and so the matrix of SEDs for the variety by nitrogen table is printed. For the first two predictions, the average SED is calculated from the average variance of differences. Note also that the order of the predictions (e.g. 0.6_cwt, 0.4_cwt 0.2_cwt 0_cwt for nitrogen) is simply the order those treatment labels were discovered in the data file.

```
Split plot analysis - oat  Variety.Nitrogen               14 Apr 2008 16:15:49
                                                          oats

Ecode is E for Estimable, * for Not Estimable

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.

---- ---- ---- ---- ---- ---- ----   1 ---- ---- ---- ---- ---- ---- ----
Predicted values of yield
The SIMPLE averaging set: variety
The ignored set: blocks wplots

nitrogen        Predicted_Value Standard_Error Ecode
0.6_cwt                123.3889         7.1747 E
0.4_cwt                114.2222         7.1747 E
0.2_cwt                 98.8889         7.1747 E
0_cwt                   79.3889         7.1747 E
SED: Overall Standard Error of Difference   4.436

---- ---- ---- ---- ---- ---- ----   2 ---- ---- ---- ---- ---- ---- ----
Predicted values of yield
The SIMPLE averaging set: nitrogen
The ignored set: blocks wplots

variety         Predicted_Value Standard_Error Ecode
Marvellous             109.7917         7.7975 E
Victory                 97.6250         7.7975 E
Golden_rain            104.5000         7.7975 E
SED: Overall Standard Error of Difference   7.079

---- ---- ---- ---- ---- ---- ----   3 ---- ---- ---- ---- ---- ---- ----
Predicted values of yield
The ignored set: blocks wplots

nitrogen        variety         Predicted_Value Standard_Error Ecode
0.6_cwt         Marvellous             126.8333         9.1070 E
0.6_cwt         Victory                118.5000         9.1070 E
0.6_cwt         Golden_rain            124.8333         9.1070 E
0.4_cwt         Marvellous             117.1667         9.1070 E
0.4_cwt         Victory                110.8333         9.1070 E
0.4_cwt         Golden_rain            114.6667         9.1070 E
0.2_cwt         Marvellous             108.5000         9.1070 E
0.2_cwt         Victory                 89.6667         9.1070 E
0.2_cwt         Golden_rain             98.5000         9.1070 E
0_cwt           Marvellous              86.6667         9.1070 E
0_cwt           Victory                 71.5000         9.1070 E
0_cwt           Golden_rain             80.0000         9.1070 E
```

```
Predicted values with SED(PV)
   126.833
   118.500        9.71503
   124.833        9.71503        9.71503
   117.167        7.68295        9.71503        9.71503
   110.833        9.71503        7.68295        9.71503        9.71503
   114.667        9.71503        9.71503        7.68295        9.71503
   9.71503
   108.500        7.68295        9.71503        9.71503        7.68295
   9.71503        9.71503
   89.6667        9.71503        7.68295        9.71503        9.71503
   7.68295        9.71503        9.71503
   98.5000        9.71503        9.71503        7.68295        9.71503
   9.71503        7.68295        9.71503        9.71503
   86.6667        7.68295        9.71503        9.71503        7.68295
   9.71503        9.71503        7.68295        9.71503        9.71503
   71.5000        9.71503        7.68295        9.71503        9.71503
   7.68295        9.71503        9.71503        7.68295        9.71503
   9.71503
   80.0000        9.71503        9.71503        7.68295        9.71503
   9.71503        7.68295        9.71503        9.71503        7.68295
   9.71503        9.71503
SED: Standard Error of Difference: Min   7.6830   Mean   9.1608   Max   9.7150
```

16.3 Unbalanced nested design - Rats

The second example we consider is a data set which illustrates some further aspects of testing fixed effects in linear mixed models. This example differs from the split plot example, as it is unbalanced and 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 in total size and in the numbers 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 experiment, which meant that one dose had only 7 dams. The analysis must account for the presence of between dam variation, but must also recognise the stratification of the experimental units (pups within litters) and that doses and littersize belong to the dam stratum. Table 16.2 presents an indicative AOV decomposition for this experiment.

Table 16.2: Rat data: AOV decomposition

stratum	decomposition	type	df or ne
constant	1	F	1
dams			
	dose	F	2
	littersize	F	1
	dam	R	27
dams.pups			
	sex	F	1
	dose.sex	F	2
error		R	

The dose and littersize effects are tested against the residual dam variation, while the remaining effects are tested against the residual within litter variation. The ASReml input to achieve this analysis is presented below.

```
Rats example
 dose 3 !A
 sex 2 !A
 littersize
 dam 27
 pup 18
 weight
rats.asd !DOPATH 1   # Change DOPATH argument to select each PATH
!PATH 1
weight ~ mu littersize dose sex dose.sex !r idv(dam)
```

```
residual idv(units)
!PATH 2
weight ~ mu out(66) littersize dose sex dose.sex !r idv(dam)
residual idv(units)
!PATH 3
weight ~ mu littersize dose sex !r idv(dam)
residual idv(units)
!PATH 4
weight ~ mu littersize dose sex
residual idv(units)
```

The input file contains an example of the use of the `!DOPATH` qualifier. Its argument specifies which part to execute. We will discuss the models in the two parts. It also includes the `!FCON` qualifier to request conditional Wald F statistics. Abbreviated output from part 1 is presented below.

```
   1 LogL= 74.2174     S2= 0.19670        315 df   0.1000      1.000
   2 LogL= 79.1579     S2= 0.18751        315 df   0.1488      1.000
   3 LogL= 83.9408     S2= 0.17755        315 df   0.2446      1.000
   4 LogL= 86.8093     S2= 0.16903        315 df   0.4254      1.000
   5 LogL= 87.2249     S2= 0.16594        315 df   0.5521      1.000
   6 LogL= 87.2398     S2= 0.16532        315 df   0.5854      1.000
   7 LogL= 87.2398     S2= 0.16530        315 df   0.5867      1.000
   8 LogL= 87.2398     S2= 0.16530        315 df   0.5867      1.000
 Final parameter values                        0.5867

          - - - Results from analysis of weight - - -
 Akaike Information Criterion     -170.48 (assuming 2 parameters).
 Bayesian Information Criterion   -162.97

          Approximate stratum variance decomposition
 Stratum     Degrees-Freedom   Variance      Component Coefficients
 idv(dam)              22.56    1.27762        11.5     1.0
 Residual Variance    292.44   0.165300         0.0     1.0

 Model_Term                             Gamma         Sigma   Sigma/SE   % C
 idv(dam)                IDV_V   27  0.586674      0.969770E-01   2.92   0 P
 idv(units)                     322 effects
 Residual                SCA_V  322  1.000000      0.165300      12.09   0 P

                                   Wald F statistics
     Source of Variation           NumDF    DenDF_con F_inc    F_con M  P_con
   7 mu                                1      32.0  9049.48  1099.20 b <.001
   3 littersize                        1      31.5    27.99    46.25 B <.001
   1 dose                              2      23.9    12.15    11.51 A <.001
   2 sex                               1     299.8    57.96    57.96 A <.001
   8 dose.sex                          2     302.1     0.40     0.40 B 0.673
 Notice: The DenDF values are calculated ignoring fixed/boundary/singular
             variance parameters using algebraic derivatives.
   4 dam                                  27 effects fitted
 SLOPES FOR LOG(ABS(RES)) on LOG(PV) for Section    1
   2.27
           3  possible outliers: see .res file
```

The iterative sequence has converged and the variance component parameter for `dam` hasn't changed for the last three iterations. The incremental Wald F statistics indicate that the interaction between `dose` and `sex` is not significant. The `F_con` column helps us to 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.

We also note the comment `3 possible outliers:  see .res file`. Checking the `.res` file, we discover unit 66 has a standardised residual of -8.80 (see Figure 16.1). 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 in the following. However, part 2 shows one way of 'dropping' unit 66 by fitting an effect for it with `out(66)`.

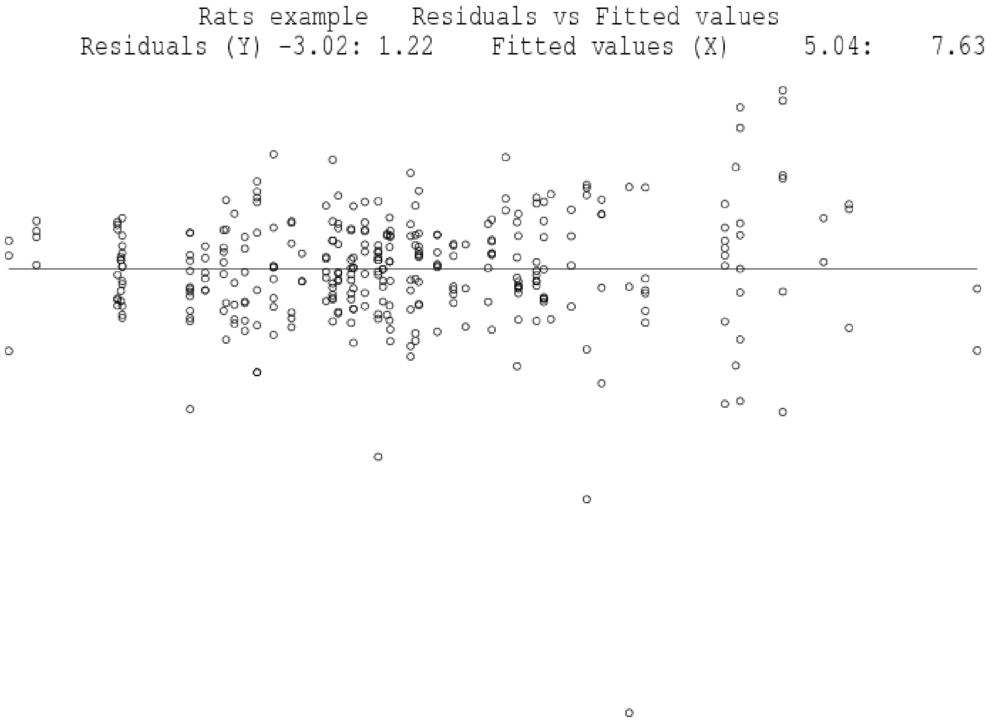

Figure 16.1: Residual plot for the rat data

We refit the model without the `dose.sex` term. Note that the variance parameters are re-estimated, though there is little change from the previous analysis.

```
Model_Term                                 Gamma         Sigma   Sigma/SE   % C
idv(dam)                 IDV_V    27   0.595157      0.979179E-01    2.93   0 P
idv(units)                       322  effects
Residual                 SCA_V   322   1.000000      0.164524       12.13   0 P

                                    Wald F statistics
    Source of Variation             NumDF     DenDF_con F_inc     F_con M P_con
```

```
   7 mu                                1      32.0  8981.48  1093.05 . <.001
   3 littersize                        1      31.4    27.85    46.43 A <.001
   1 dose                              2      24.0    12.05    11.42 A <.001
   2 sex                               1     301.7    58.27    58.27 A <.001
```

Part 4 shows what happens if we (wrongly) drop `dam` from this model. Even if a random term is not 'significant', it should not be dropped from the model when we are testing fixed effects, or desire standard errors of adjusted means, if it represents a strata of the design as in this case.

```
 Model_Term                             Gamma         Sigma   Sigma/SE   % C
 idv(units)                     322 effects
 Residual                SCA_V  322  1.000000      0.253182      12.59   0 P

                                   Wald F statistics
     Source of Variation           NumDF    DenDF_con F_inc    F_con M P_con
   7 mu                                1     317.0 47077.31  3309.42 . <.001
   3 littersize                        1     317.0    68.48   146.50 A <.001
   1 dose                              2     317.0    60.99    58.43 A <.001
   2 sex                               1     317.0    24.52    24.52 A <.001
```

16.4 Source of variability in unbalanced data - Volts

In this example we illustrate an analysis of unbalanced data in which 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. Regulators are taken from the production line to a setting station and adjusted to operate within a specified voltage range. From the setting station the regulator is then passed to a testing station where it is tested and returned if outside the required range.

The voltage of 64 regulators was set at 10 setting stations (`setstat`); between 4 and 8 regulators were set at each station. The regulators were each tested at four testing stations (`teststat`). The ASReml input file is presented below.

```
Voltage data
 teststat 4   # 4 testing stations tested each regulator
 setstat  !A  # 10 setting stations each set 4-8 regulators
 regulator 8   # regulators numbered within setting stations
 voltage
voltage.asd !skip 1
voltage ~ mu !r idv(setstat) idv(setstat.regulator) idv(teststat) idv(setstat.teststat)
residual idv(units)
```

The factor `regulator` numbers the regulators within each setting station. Thus the term `setstat.regulator` fits an effect for each regulator, while the other terms examine the effects of the setting and testing stations and possible interaction. The abbreviated output is given below

```
LogL= 188.604     S2= 0.67074E-01    255 df
LogL= 199.530     S2= 0.59303E-01    255 df
LogL= 203.007     S2= 0.52814E-01    255 df
LogL= 203.240     S2= 0.51278E-01    255 df
LogL= 203.242     S2= 0.51141E-01    255 df
LogL= 203.242     S2= 0.51140E-01    255 df

Model_Term                             Gamma         Sigma   Sigma/SE   % C
idv(TestStat)           IDV_V    4   0.642752E-01  0.328704E-02    0.98   0 P
idv(Setstat)            IDV_V   10   0.233416      0.119369E-01    1.35   0 P
idv(TestStat.Setstat)   IDV_V   40   0.101193E-06  0.517501E-08    0.00   0 B
idv(Regulator.Setstat)  IDV_V   80   0.601817      0.307770E-01    3.64   0 P
idv(units)                     256 effects
Residual                SCA_V  256   1.000000      0.511400E-01    9.72   0 P
Warning: Code B - fixed at a boundary (!GP)       F - fixed by user
              ? - liable to change from P to B    P - positive definite
              C - Constrained by user (!VCC)      U - unbounded
              S - Singular Information matrix
```

The convergence criteria has been satisfied after six iterations. A warning message is printed below the summary of the variance components because the variance component for the `setstat.teststat` term has been fixed near the boundary. The default constraint for variance components (`!GP`) is to ensure that the REML estimate remains positive. Under this constraint, if an update for any variance component results in a negative value then ASReml sets that variance component to a small positive value. If this occurs in subsequent iterations the parameter is fixed to a small positive value and the code `B` replaces `P` in the `C` column of the summary table. The default constraint can be overridden using the `!GU` qualifier, but it is not generally recommended for standard analyses.

Figure 16.2 presents the residual plot which indicates two unusual data values. These values are successive observations, namely observation 210 and 211, being testing stations 2 and 3 for setting station 9(J), regulator 2. These observations will not be dropped from the following analyses for consistency with other analyses conducted by Cox and Snell (1981) and in the GENSTAT manual.

The REML log-likelihood from the model without the `setstat.teststat` term was 203.242, the same as the REML log-likelihood for the previous model. Table 16.3 presents a summary of the REML log-likelihood ratio for the remaining terms in the model. The summary of the ASReml output for the current model is given below. The column labelled `Sigma/SE` is printed by ASReml to give a guide as to the significance of the variance component for each term in the model. The statistic is simply the REML estimate of the variance component divided by the square root of the diagonal element (for each component) of the inverse of the average information matrix. The diagonal elements of the expected (not the average) information matrix are the asymptotic variances of the REML estimates of the variance parameters. These `Sigma/SE` statistics cannot be used to test the null hypothesis that the variance component is zero. If we had used this crude measure then the conclusions would have been inconsistent with the conclusions obtained from the REML log-likelihood ratio test

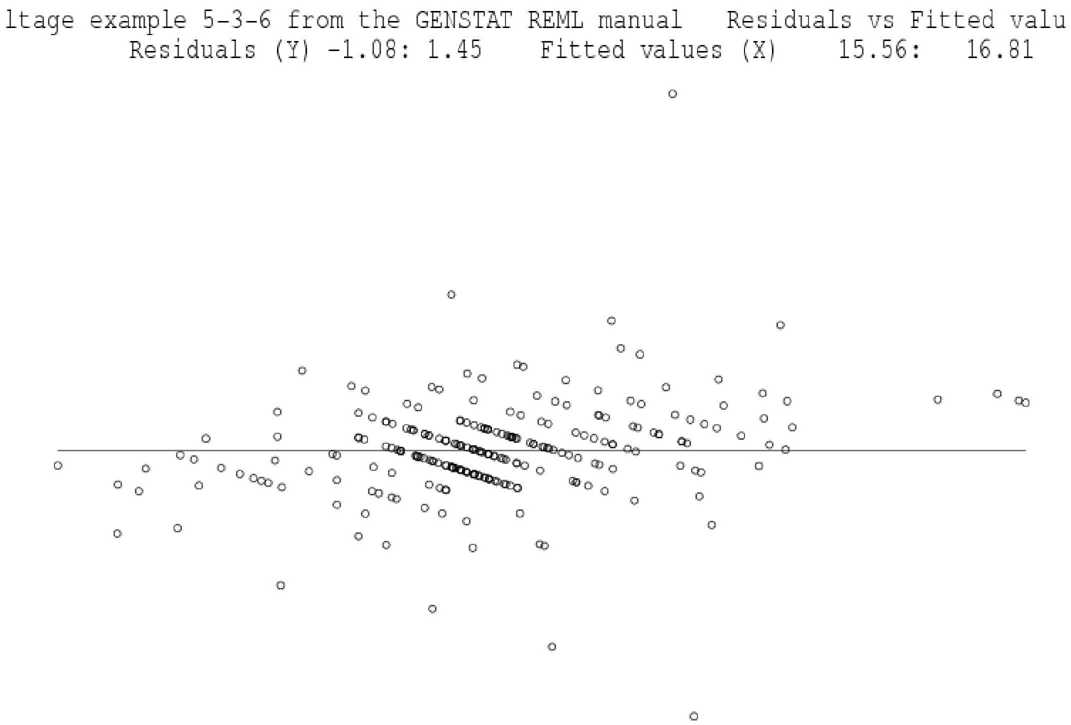

Figure 16.2: Residual plot for the voltage data

(see Table 16.3).

```
Model_Term                                Gamma         Sigma   Sigma/SE   % C
idv(TestStat)          IDV_V     4  0.642752E-01  0.328704E-02    0.98    0 P
idv(Setstat)           IDV_V    10  0.233416      0.119369E-01    1.35    0 P
idv(Regulator.Set)     IDV_V    80  0.601817      0.307770E-01    3.64    0 P
idv(units)                     256 effects
Residual               SCA_V   256  1.000000      0.511400E-01    9.72    0 P
```

Table 16.3: REML log-likelihood ratio for the variance components in the voltage data

	REML	−2×	
terms	log-likelihood	difference	P-value
− `setstat`	200.31	5.864	.0077
− `setstat.regulator`	184.15	38.19	.0000
− `teststat`	199.71	7.064	.0039

16.5 **Balanced repeated measures** - Height

The data for this example is taken from the GENSTAT manual. It consists of a total of 5 measurements of height (cm) taken on 14 plants. The 14 plants were either diseased or healthy and were arranged in a glasshouse in a completely random design. The heights were measured 1, 3, 5, 7 and 10 weeks after the plants were placed in the glasshouse. There were 7 plants in each treatment. The data are depicted in Figure 16.3 obtained by qualifier line
`!Y y1 !G tmt !JOIN`
in the following multivariate ASReml job.

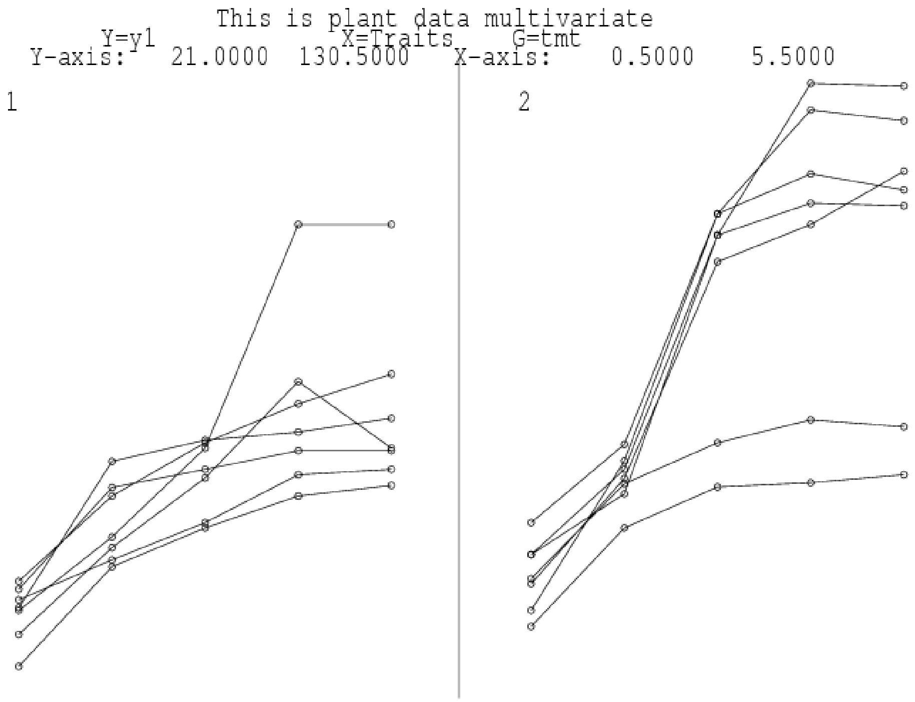

Figure 16.3: Trellis plot of the height for each of 14 plants

In the following we illustrate how various repeated measures analyses can be conducted in ASReml. For these analyses it is convenient to arrange the data in a multivariate form, with 7 fields representing the plant number, treatment identification and the 5 heights. The ASReml input file for our first model is

```
This is plant data multivariate
 tmt   !A  # Diseased Healthy
 plant 14
 y1 y3 y5 y7 y10
grass.asd !skip 1 !ASUV
 !Y y1 !G tmt !JOIN  # Plot the data
y1 y3 y5 y7 y10 ~ Trait tmt Tr.tmt !r idv(units)
residual idv(units.Trait)
```

The focus is modelling of the error variance for the data. Specifically we fit the multivariate regression model given by

$$\boldsymbol{Y} = \boldsymbol{DT} + \boldsymbol{E} \tag{16.1}$$

where $\boldsymbol{Y}^{14\times5}$ is the matrix of heights, $\boldsymbol{D}^{14\times2}$ is the design matrix, $\boldsymbol{T}^{2\times5}$ is the matrix of fixed effects and $\boldsymbol{E}^{14\times5}$ is the matrix of errors. The heights taken on the same plants will be correlated and so we assume that

$$\text{var}\left(\text{vec}(\boldsymbol{E})\right) = \boldsymbol{I}_{14} \otimes \boldsymbol{\Sigma} \tag{16.2}$$

where $\boldsymbol{\Sigma}^{5\times5}$ is a symmetric positive definite matrix.

The variance models used for $\boldsymbol{\Sigma}$ are given in Table 16.4. These represent some commonly used models for the analysis of repeated measures data (see Wolfinger, 1986). Note that we have specified the `!ASUV` qualifier. This is required to allow the fitting of all these models. Without `!ASUV`, ASReml woul only allow us to fit the final (UnStructured) variance model which is the default $\boldsymbol{R}$ structure fo

Table 16.4: Summary of variance models fitted to the plant data

model	number of parameters	REML log-likelihood	BIC
Uniform	2	-196.88	401.95
Power	2	-182.98	374.15
Heterogeneous Power	6	-171.50	367.57
Antedependence (order 1)	9	-160.37	357.51
Unstructured	15	-158.04	377.50

The split plot in time model can be fitted in two ways, either by fitting a `units` term plus an ind residual as above, or by specifying a `CORU` variance model for the R-structure as follows

```
y1 y3 y5 y7 y10 ~ Trait tmt Tr.tmt
residual id(units).coru(Trait)
```

The two forms for Σ are given by

$$\begin{array}{rcll} \boldsymbol{\Sigma} &=& \sigma_1^2\boldsymbol{J} + \sigma_2^2\boldsymbol{I}, & \texttt{units} \\ \boldsymbol{\Sigma} &=& \sigma_e^2\boldsymbol{I} + \sigma_e^2\rho(\boldsymbol{J}-\boldsymbol{I}), & \texttt{CORU} \end{array} \tag{16.3}$$

It follows that

$$\begin{array}{rcl} \sigma_e^2 &=& \sigma_1^2 + \sigma_2^2 \\ \rho &=& \frac{\sigma_1^2}{\sigma_1^2+\sigma_2^2} \end{array} \tag{16.4}$$

Portions of the two outputs are given below. The REML log-likelihoods for the two models are the same and it is easy to verify that the REML estimates of the variance parameters satisfy (16.4), viz. $\sigma_e^2 = 286.310 \approx 159.858 + 126.528 = 286.386$; $159.858/286.386 = 0.558191$.

```
#
# !r idv(units.Trait)
#
 LogL=-204.593     S2=  224.61         60 df   0.1000      1.000
 LogL=-201.233     S2=  186.52         60 df   0.2339      1.000
 LogL=-198.453     S2=  155.09         60 df   0.4870      1.000
 LogL=-197.041     S2=  133.85         60 df   0.9339      1.000
 LogL=-196.881     S2=  127.56         60 df    1.204      1.000
 LogL=-196.877     S2=  126.53         60 df    1.261      1.000
 Final parameter values                          1.2634     1.0000
          - - - Results from analysis of y1 y3 y5 y7 y10 - -
 Akaike Information Criterion     397.75 (assuming 2 parameters)
 Bayesian Information Criterion    401.9
          Approximate stratum variance decomposition
 Stratum     Degrees-Freedom   Variance      Component Coefficients
 idv(units)            12.00    925.584          5.0     1.0
 Residual Variance     48.00    126.494          0.0     1.0

 Model_Term                             Gamma         Sigma   Sigma/SE   % C
 idv(units)                IDV_V   14   1.26342       159.816       2.11   0 P
 idv(units.Trait)                  70 effects
 Residual                  SCA_V   70  1.000000       126.494       4.90   0 P
#
# id(units).coru(Trait)
#
 LogL=-196.975     S2=  264.10         60 df    1.000     0.5000
 LogL=-196.924     S2=  270.14         60 df    1.000     0.5178
 LogL=-196.886     S2=  278.58         60 df    1.000     0.5400
 LogL=-196.877     S2=  286.23         60 df    1.000     0.5580
 LogL=-196.877     S2=  286.31         60 df    1.000     0.5582
 Final parameter values                          1.0000    0.55819
          - - - Results from analysis of y1 y3 y5 y7 y10 - -
 Akaike Information Criterion     397.75 (assuming 2 parameters)
 Bayesian Information Criterion    401.9

 Model_Term                             Gamma         Sigma   Sigma/SE   % C
 id(units).coru(Trait)             70 effects
 Residual                  SCA_V   70  1.000000       286.310       3.65   0 P
 Trait                        COR_R    1  0.558191      0.558191       4.28   0 P
```

A more realistic model for repeated measures data would allow the correlations to decrease as the lag increases such as occurs with the first order autoregressive model. However, since the heights are not measured at equally spaced time points we use the EXP model. The correlation function is given by

$$\rho(u) = \phi^u$$

where u is the time lag is weeks. The coding for this is

```
y1 y3 y5 y7 y10 ~ Trait tmt Tr.tmt
residual id(units).exp(Trait !INIT 0.5 !COORD 1 3 5 7 10 )
```

A portion of the output is

```
  1 LogL=-202.139     S2=  234.04         60 df   1.0000     0.5000
  2 LogL=-183.773     S2=  440.42         60 df   1.0000     0.9507
  3 LogL=-183.070     S2=  337.51         60 df   1.0000     0.9308
  4 LogL=-182.981     S2=  297.16         60 df   1.0000     0.9172
  5 LogL=-182.979     S2=  302.31         60 df   1.0000     0.9193
  6 LogL=-182.979     S2=  301.45         60 df   1.0000     0.9190
 Final parameter values                          1.0000     0.9190

           - - - Results from analysis of y1 y3 y5 y7 y10 - - -
 Akaike Information Criterion      369.96 (assuming 2 parameters).
 Bayesian Information Criterion    374.15

 Model_Term                              Gamma         Sigma   Sigma/SE   % C
 id(units).exp(Trait)             70 effects
 Residual                 SCA_V   70  1.000000       301.449       3.12   0 P
 Trait                    EXP_P    1  0.919007      0.919007      29.49   0 P
```

When fitting power models be careful to ensure the scale of the defining variate, here `time`, 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 .993.

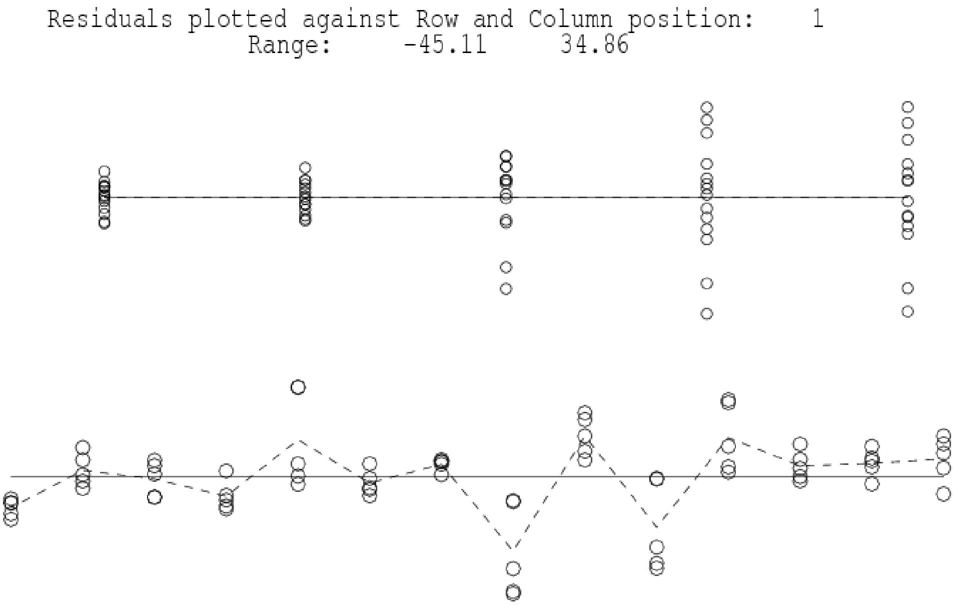

Figure 16.4: Residual plots for the `EXP` variance model for the plant data

The residual plot from this analysis is presented in Figure 16.4. This suggests increasing variance over time. This can be modelled by using the `EXPH` model, which models Σ 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 coding for this is

```
y1 y3 y5 y7 y10 ~ Trait tmt Tr.tmt
residual id(units).exph(Trait !INIT 0.5 100 200 300 300 300 !COORD 1 3 5 7 10)
```

Abbreviated output from this analysis is

```
  9 LogL=-171.512     S2= 1.00000         60 df
 10 LogL=-171.500     S2= 1.00000         60 df
 11 LogL=-171.497     S2= 1.00000         60 df
 12 LogL=-171.496     S2= 1.00000         60 df

           - - - Results from analysis of y1 y3 y5 y7 y10 - - -
 Akaike Information Criterion      354.99 (assuming 6 parameters).
 Bayesian Information Criterion    367.56

 Model_Term                              Sigma         Sigma   Sigma/SE   % C
 id(units).exph(Trait)             70 effects
 Trait                   EXP_P    1   0.906843      0.906843      21.88   0 P
 Trait                   EXP_V    1    60.8955       60.8955       2.12   0 P
 Trait                   EXP_V    2    73.0128       73.0128       1.99   0 P
 Trait                   EXP_V    3    309.013       309.013       2.22   0 P
 Trait                   EXP_V    4    435.964       435.964       2.52   0 P
 Trait                   EXP_V    5    382.312       382.312       2.74   0 P
 Covariance/Variance/Correlation Matrix  Residual
   61.05      0.8227      0.6768      0.5568      0.4155
   54.90       72.95      0.8227      0.6768      0.5050
   93.05       123.6       309.6      0.8227      0.6139
   90.95       120.8       302.6       437.0      0.7462
   63.49       84.36       211.2       305.1       382.5
                            Wald F statistic
    Source of Variation     NumDF      DenDF     F-inc          P-in
  8 Trait                       5       18.7    108.25          <.00
  1 tmt                         1       13.1      0.00          0.96
  9 Tr.tmt                      4       21.0      4.37          0.01
```

The last two models we fit are the antedependence model of order 1 and the unstructured model. Starting values need not actually be supplie in this example (the defaults are adequate) but are suppled t demonstrate the syntax. We use the REML estimate of $\boldsymbol{\Sigma}$ from the heterogeneous power model shown in the previous output. The antedependence model models $\boldsymbol{\Sigma}$ by the inverse cholesky decomposition

$$\boldsymbol{\Sigma}^{-1} = \boldsymbol{U}\boldsymbol{D}\boldsymbol{U}'$$

where $\boldsymbol{D}$ is a diagonal matrix and $\boldsymbol{U}$ is a unit upper triangular matrix. For an antedependence model of order q, then $u_{ij} = 0$ for $j > i + q - 1$. The antedependence model of order 1 has 9 parameters for these data, 5 in $\boldsymbol{D}$ and 4 in $\boldsymbol{U}$. The input is given by

```
!ASSIGN ANTEI !<  !INIT
 60.1
 54.65 73.65
 91.50 123.3 306.4
 89.17 120.2 298.6 431.8
 62.21 83.85 208.3 301.2 379.8
 !>
y1 y3 y5 y7 y10 ~ Trait tmt Tr.tmt
redidual units.ante(Trait $ANTEI)
```

The abbreviated output file is

```
 1 LogL=-171.501     S2=  1.0000         60 df
 2 LogL=-170.097     S2=  1.0000         60 df
 3 LogL=-166.085     S2=  1.0000         60 df
 4 LogL=-161.335     S2=  1.0000         60 df
 5 LogL=-160.407     S2=  1.0000         60 df
 6 LogL=-160.370     S2=  1.0000         60 df
 7 LogL=-160.369     S2=  1.0000         60 df
 8 LogL=-160.369     S2=  1.0000         60 df
 9 LogL=-160.369     S2=  1.0000         60 df
          - - - Results from analysis of y1 y3 y5 y7 y10 - -
 Akaike Information Criterion     338.74 (assuming 9 parameters)
 Bayesian Information Criterion    357.59

 Model_Term                              Sigma         Sigma   Sigma/SE   % C
 id(units).ante(Trait)            70 effects
 Trait                 ANTE_U  1  1  0.268643E-01  0.268643E-01    2.44    0 P
 Trait                 ANTE_U  2  1 -0.628417     -0.628417      -2.55    0 P
 Trait                 ANTE_U  2  2  0.372830E-01  0.372830E-01    2.41    0 P
 Trait                 ANTE_U  3  2  -1.49102      -1.49102      -2.54    0 P
 Trait                 ANTE_U  3  3  0.599612E-02  0.599612E-02    2.43    0 P
 Trait                 ANTE_U  4  3  -1.28037      -1.28037      -6.19    0 P
 Trait                 ANTE_U  4  4  0.789716E-02  0.789716E-02    2.44    0 P
 Trait                 ANTE_U  5  4 -0.967820     -0.967820     -15.40    0 P
 Trait                 ANTE_U  5  5  0.390635E-01  0.390635E-01    2.45    0 P
 Covariance/Variance/Correlation Matrix ANTE Residual
   37.20      0.5946      0.3550      0.3115      0.3041
   23.38       41.55      0.5970      0.5239      0.5114
   34.84       61.93       258.9      0.8776      0.8566
   44.60       79.27       331.5       550.9      0.9761
   43.16       76.72       320.8       533.2       541.6

                                 Wald F statistics
     Source of Variation         NumDF              F-inc
   8 Trait                           5             188.83
   1 tmt                             1               4.14
   9 Trait.tmt                       4               3.91
```

The iterative sequence converged and the antedependence parameter estimates are printed columnwise by time, the column of $\boldsymbol{U}$ and the element of $\boldsymbol{D}$. I.e.

$$\boldsymbol{D} = \text{diag}\begin{bmatrix} 0.0269 \\ 0.0373 \\ 0.0060 \\ 0.0079 \\ 0.0391 \end{bmatrix}, \boldsymbol{U} = \begin{bmatrix} 1 & -0.6284 & 0 & 0 & 0 \\ 0 & 1 & -1.4911 & 0 & 0 \\ 0 & 0 & 1 & -1.2804 & 0 \\ 0 & 0 & 0 & 1 & -0.9678 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

Finally the input and output files for the unstructured model are presented below. The REML estimate of Σ from the ANTE model is used to provide starting values.

```
!ASSIGN USI !< !INIT
  37.20
  23.38      41.55
  34.83      61.89      258.9
  44.58      79.22      331.4      550.8
  43.14      76.67      320.7      533.0      541.4
  !>
y1 y3 y5 y7 y10 ~ Trait tmt Trait.tmt
residual id(units).us(Trait $USI)
```

```
1 LogL=-160.368     S2=  1.0000         60 df
2 LogL=-159.027     S2=  1.0000         60 df
3 LogL=-158.247     S2=  1.0000         60 df
4 LogL=-158.040     S2=  1.0000         60 df
5 LogL=-158.036     S2=  1.0000         60 df

         - - - Results from analysis of y1 y3 y5 y7 y10 - -
 Akaike Information Criterion     346.07 (assuming 15 parameters)
 Bayesian Information Criterion    377.49

 Model_Term                               Sigma         Sigma   Sigma/SE   % C
 id(units).us(Trait)               70 effects
 Trait                    US_V  1  1    37.2262       37.2262       2.45   0 P
 Trait                    US_C  2  1    23.3935       23.3935       1.77   0 P
 Trait                    US_V  2  2    41.5195       41.5195       2.45   0 P
 Trait                    US_C  3  1    51.6524       51.6524       1.61   0 P
 Trait                    US_C  3  2    61.9169       61.9169       1.78   0 P
 Trait                    US_V  3  3    259.121       259.121       2.45   0 P
 Trait                    US_C  4  1    70.8113       70.8113       1.54   0 P
 Trait                    US_C  4  2    57.6146       57.6146       1.23   0 P
 Trait                    US_C  4  3    331.807       331.807       2.29   0 P
 Trait                    US_V  4  4    551.507       551.507       2.45   0 P
 Trait                    US_C  5  1    73.7857       73.7857       1.60   0 P
 Trait                    US_C  5  2    62.5691       62.5691       1.33   0 P
 Trait                    US_C  5  3    330.851       330.851       2.29   0 P
 Trait                    US_C  5  4    533.756       533.756       2.42   0 P
 Trait                    US_V  5  5    542.175       542.175       2.45   0 P
```

However, the usual syntax for fitting an unstructured error model for multivariate data is to omit the `!ASUV` qualifier and write

```
y1 y3 y5 y7 y10 ~ Trait tmt Tr.tmt
residual  id(units).us(Trait)
```

The antedependence model of order 1 is clearly more parsimonious than the unstructured model. Table 16.5 presents the incremental Wald F statistics for each of the variance models. There is a surprising level of discrepancy between models for the Wald F statistics. The main effect of treatment is significant for the uniform, power and antedependence models.

Table 16.5: Summary of Wald F statistics for fixed effects for variance models fitted to the plan

	treatment	treatment.time
model	(df=1)	(df=4)
Uniform	9.41	5.10
Power	6.86	6.13
Heterogeneous power	0.00	4.81
Antedependence (order 1)	4.14	3.91
Unstructured	1.71	4.46

16.6 Spatial analysis of a field experiment - Barley

In this section we illustrate the ASReml syntax for performing spatial and incomplete block analysis of a field experiment. 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 (see for example, Cullis and Gleeson, 1991; Cressie, 1991; Gilmour *et al.*, 1997). This example illustrates the analysis of 'so-called' regular spatial data, in which the data is observed on a lattice or regular grid. This is typical of most small plot designed field experiments. Spatial data is often irregularly spaced, either by design or because of the observational nature of the study. The techniques we present in the following can be extended for the analysis of irregularly spaced spatial data, though, larger spatial data sets may be computationally challenging, depending on the degree of irregularity or models fitted.

The data we consider is taken from Gilmour *et al.* (1995) and involves 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 replicates laid out as shown in Table 16.6. The data fields were `Rep, RowBlk, ColBlk, row, column` and `yield`. Lattice row and column numbering is typically within replicates and so the terms specified in the linear model to account for the lattice row and lattice column effects would be `Rep.latticerow Rep.latticecolumn`. However, in this example lattice rows and columns are both numbered from 1 to 30 across replicates (see Table 16.6). The terms in the linear model are therefore simply `RowBlk ColBlk`. Additional fields `row` and `column` indicate the spatial layout of the plots.

The ASReml input file is presented below. Three models have been fitted to these data. The lattice analysis is included for comparison in `PATH 3`. In `PATH 1` we use the separable first order autoregressive model to model 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 (R structure) is given by

$$\sigma^2\boldsymbol{\Sigma} = \sigma^2(\boldsymbol{\Sigma}_c \otimes \boldsymbol{\Sigma}_r) \tag{16.5}$$

where $\boldsymbol{\Sigma}_c$ and $\boldsymbol{\Sigma}_r$ are 15×15 and 10×10 matrix functions of the column (ϕ_c) and row (ϕ_r) autoregressive parameters respectively. 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 (from the current model). This diagnostic and a summary of row and column residual trends are produced by default with graphical versions of ASReml when a spatial model has been fitted to the errors. It can be suppressed, by the use of the `-n` option on the command line. We have produced the following plots by use of the `!EPS` qualifier. The `!RENAME !ARG 1 2 3` qualifiers in conjunctio with `!DOPART $1` cause ASReml to run all three parts, appending the part number to the output file names.

Table 16.6: 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

```
!EPS !RENAME !ARG 1 2
Slate Hall example
  Rep 6      # Six replicates of 5x5 plots in 2x3 arrangement
  RowBlk 30  # Rows within replicates numbered across replicates
  ColBlk 30  # Columns within replicates numbered across replicates
  row 10     # Field row
  column 15  # Field column
  variety 25
  yield
barley.asd !skip 1 !DOPATH $1
!PATH 1 # AR1 x AR1
y ~ mu var
residual ar1v(column).ar1(row)
!PATH 2 # AR1 x AR1 + units
y ~ mu var !r idv(units)
residual ar1v(column).ar1(row)
!PATH 3 # incomplete blocks
y ~ mu var !r idv(Rep) idv(Rowblk) idv(Colblk)
residual idv(units)
!PATH 0
predict variety !TWOSTAGEWEIGHTS
```

Abbreviated ASReml output file is presented below. The iterative sequence has converged to column and row correlation parameters of (.68377,.45859) 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. Figure 16.5 presents a plot of the sample variogram of the residuals from this model. The plot appears in reasonable agreement with the model.

The next model includes a measurement error or nugget effect component. That is the variance model for the plot errors is now given by

$$\sigma^2\mathbf{\Sigma} = \sigma^2(\mathbf{\Sigma}_c \otimes \mathbf{\Sigma}_r) + \psi \boldsymbol{I}_{150} \tag{16.6}$$

where ψ is the ratio of nugget variance to error variance (σ^2). The abbreviated output for this model is given below. There is a significant improvement in the REML log-likelihood with the inclusion of the nugget effect (see Table 16.7).

```
# AR1 x AR1
#
   1 LogL=-739.681     S2=  36034.        125 df    1.000     0.1000      0.1000
   2 LogL=-714.340     S2=  28109.        125 df    1.000     0.4049      0.1870
   3 LogL=-703.338     S2=  29914.        125 df    1.000     0.5737      0.3122
   4 LogL=-700.371     S2=  37464.        125 df    1.000     0.6789      0.4320
   5 LogL=-700.324     S2=  38602.        125 df    1.000     0.6838      0.4542
   6 LogL=-700.322     S2=  38735.        125 df    1.000     0.6838      0.4579
   7 LogL=-700.322     S2=  38754.        125 df    1.000     0.6838      0.4585
   8 LogL=-700.322     S2=  38757.        125 df    1.000     0.6838      0.4586
 Final parameter values                        1.0000    0.68377     0.45861

          - - - Results from analysis of yield - - -
 Akaike Information Criterion     1406.64 (assuming 3 parameters).
 Bayesian Information Criterion   1415.13
```

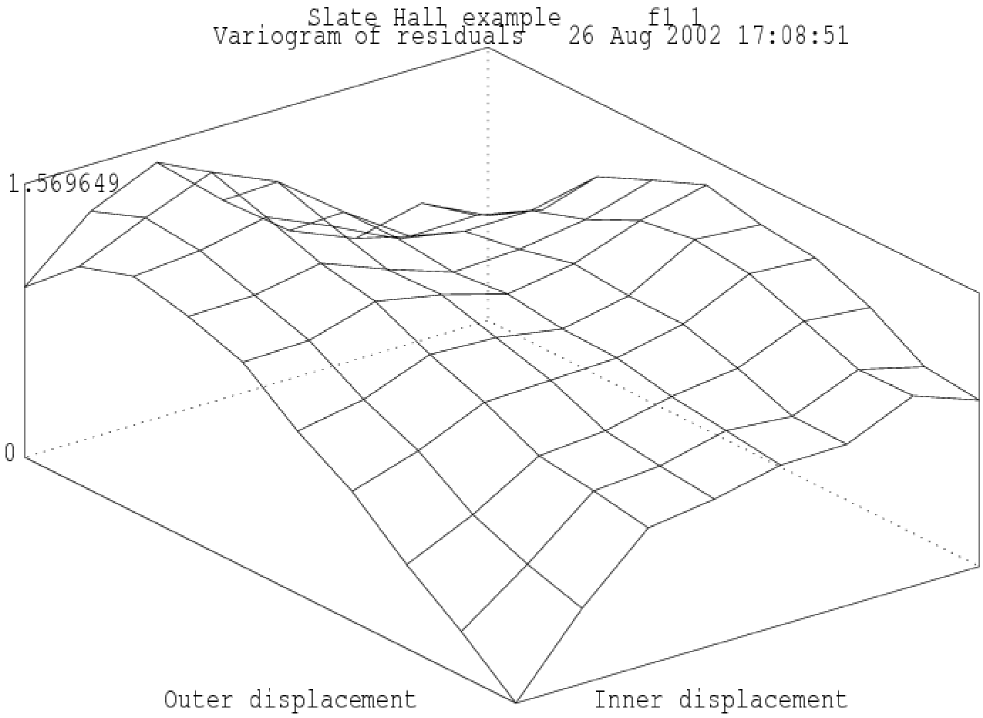

Figure 16.5: Sample variogram of the residuals from the AR1×AR1 model

```
Model_Term                               Gamma         Sigma   Sigma/SE   % C
ar1(column).ar1(row)                150 effects
Residual                SCA_V   150   1.000000       38754.3       5.00   0 P
column                   AR_R     1   0.683769      0.683769      10.80   0 P
row                      AR_R     1   0.458594      0.458594       5.55   0 P

                                  Wald F statistics
    Source of Variation           NumDF     DenDF     F_inc             Prob
  8 mu                                1      12.8    850.88             <.001
  6 variety                          24      80.0     13.04             <.001

# AR1 x AR1 + units
  1 LogL=-740.735     S2=  33225.       125 df    :   2 components constrained
  2 LogL=-723.595     S2=  11661.       125 df    :   1 components constrained
  3 LogL=-698.498     S2=  46239.       125 df
  4 LogL=-696.847     S2=  44725.       125 df
  5 LogL=-696.823     S2=  45563.       125 df
  6 LogL=-696.823     S2=  45753.       125 df
  7 LogL=-696.823     S2=  45796.       125 df

          - - - Results from analysis of yield - - -
 Akaike Information Criterion     1401.65 (assuming 4 parameters).
 Bayesian Information Criterion   1412.96

 Model_Term                               Gamma         Sigma   Sigma/SE   % C
 idv(units)             IDV_V   150   0.106152       4861.06       2.72   0 P
 ar1(column).ar1(row)           150 effects
 Residual                SCA_V  150  1.000000      45793.4       2.74   0 P
 column                   AR_R    1  0.843791     0.843791      12.33   0 P
 row                      AR_R    1  0.682682     0.682682       6.68   0 P

                                   Wald F statistics
     Source of Variation           NumDF     DenDF    F-inc           P-inc
   8 mu                                1       3.5   259.83           <.001
   6 variety                          24      75.7    10.21           <.001
```

The lattice analysis (with recovery of between block information) is presented below. This variance model is not competitive with the preceding spatial models. The models can be formally compared using the BIC values for example.

```
# IB analysis
   1 LogL=-734.184     S2=  26778.        125 df
   2 LogL=-720.060     S2=  16591.        125 df
   3 LogL=-711.119     S2=  11173.        125 df
   4 LogL=-707.937     S2=  8562.4        125 df
   5 LogL=-707.786     S2=  8091.2        125 df
   6 LogL=-707.786     S2=  8061.8        125 df
   7 LogL=-707.786     S2=  8061.8        125 df

          - - - Results from analysis of yield - - -
 Akaike Information Criterion     1423.57 (assuming 4 parameters).
 Bayesian Information Criterion   1434.88

          Approximate statum variance decomposition
 Stratum     Degrees-Freedom  Variance      Component Coefficients
 idv(Rep)               5.00    266657.       25.0     5.0     5.0     1.0
 idv(RowBlk)           24.00    74887.8        0.0     4.3     0.0     1.0
 idv(ColBlk)           23.66    71353.5        0.0     0.0     4.3     1.0
 Residual Variance     72.34    8061.81        0.0     0.0     0.0     1.0

 Model_Term                             Gamma         Sigma   Sigma/SE   % C
 idv(Rep)                IDV_V    6  0.528714      4262.39       0.62   0 P
 idv(RowBlk)             IDV_V   30   1.93444      15595.1       3.06   0 P
 idv(ColBlk)             IDV_V   30   1.83725      14811.6       3.04   0 P
 idv(units)                     150 effects
 Residual                SCA_V  150  1.000000      8061.81       6.01   0 P

                                   Wald F statistics
     Source of Variation           NumDF     DenDF    F_inc            Prob
   8 mu                                1       5.0  1216.29           <.001
   6 variety                          24      79.3     8.84           <.001
```

Finally, we present portions of the `.pvs` files to illustrate the prediction facility of ASReml. The first five and last three variety means are presented for illustration. The overall SED printed is the square root of the average variance of difference between the variety means. The two spatial analyses have a range of SEDs which are available if the `!SED` qualifier is used. All variety comparisons have the same SED from the third analysis as the design is a balanced lattice square. The Wald F statistic statistics for the spatial models are greater than for the lattice analysis. We note the Wald F statistic for the `AR1`×`AR1 + units` model is smaller than the Wald F statistic for the `AR1`×`AR1`.

```
Predicted values of yield
#AR1 x AR1
 variety             Predicted_Value Standard_Error Ecode
       1.0000              1257.9763        64.6146 E
       2.0000              1501.4483        64.9783 E
       3.0000              1404.9874        64.6260 E
       4.0000              1412.5674        64.9027 E
       5.0000              1514.4764        65.5889 E
         .                     .                .
      23.0000              1311.4888        64.0767 E
      24.0000              1586.7840        64.7043 E
      25.0000              1592.0204        63.5939 E
 SED: Overall Standard Error of Difference   59.05

#AR1 x AR1 + units
 variety             Predicted_Value Standard_Error Ecode
       1.0000              1245.5843        97.8591 E
       2.0000              1516.2331        97.8473 E
       3.0000              1403.9863        98.2398 E
       4.0000              1404.9202        97.9875 E
       5.0000              1471.6197        98.3607 E
         .                     .                .
      23.0000              1316.8726        98.0402 E
      24.0000              1557.5278        98.1272 E
      25.0000              1573.8920        97.9803 E
 SED: Overall Standard Error of Difference   60.51

# IB
 Rep                  is ignored in the prediction
 RowBlk               is ignored in the prediction
 ColBlk               is ignored in the prediction

 variety             Predicted_Value Standard_Error Ecode
       1.0000              1283.5870        60.1994 E
       2.0000              1549.0133        60.1994 E
       3.0000              1420.9307        60.1994 E
       4.0000              1451.8554        60.1994 E
       5.0000              1533.2749        60.1994 E
         .                     .                .
      23.0000              1329.1088        60.1994 E
      24.0000              1546.4699        60.1994 E
      25.0000              1630.6285        60.1994 E
 SED: Overall Standard Error of Difference   62.02
```

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

The `predict` statement included the qualifier `!TWOSTAGEWEIGHTS`. This generates an extra table in the `.pvs` file which we now display for each model.

Table 16.7: Summary of models for the Slate Hall data

model	REML log-likelihood	number of parameters	Wald F statistic	SED
AR1×AR1	-700.32	3	13.04	59.0
AR1×AR1 + units	-696.82	4	10.22	60.5
IB	-707.79	4	8.84	62.0

```
Predicted values with Effective Replication assuming
Variance= 38754.26
Heron:     1  1257.98      22.1504
Heron:     2  1501.45      20.6831
Heron:     3  1404.99      22.5286
Heron:     4  1412.57      22.7623
Heron:     5  1514.48      21.1830
   .       .      .           .
Heron:    25  1592.02      26.0990

Predicted values with Effective Replication assuming
Variance= 45796.58
Heron:     1  1245.58      23.8842
Heron:     2  1516.24      22.4423
Heron:     3  1403.99      24.1931
Heron:     4  1404.92      24.0811
Heron:     5  1471.61      23.2995
   .       .      .           .
Heron:    25  1573.89      26.0505

Predicted values with Effective Replication assuming
Variance= 8061.808
Heron:     1  1283.59      4.03145
Heron:     2  1549.01      4.03145
Heron:     3  1420.93      4.03145
Heron:     4  1451.86      4.03145
Heron:     5  1533.27      4.03145
   .       .      .          .
Heron:    25  1630.63      4.03145
```

The value of 4 for the IB analysis is clearly reasonable given there are 6 actual replicates but this analysis has used up 48 degrees of freedom for the `rowblk` and `colblk` effects. The precision from the spatial analyse (45796.58/23.8842 = 1917.442 *c.f.* 8061.808/4.03145= 1999.729) are similar but slightly lower reflecting the gain in accuracy from the spatial analysis. For further reading, see Smith *et al.* (2001, 2005).

16.7 Unreplicated early generation variety trial - Wheat

To further illustrate the approaches presented in the previous section, we consider an unreplicated field experiment conducted at Tullibigeal situated in south-western NSW. The trial was an S1 (early stage) wheat variety evaluation trial and consisted of 525 test lines which were randomly assigned to plots in a 67 by 10 array. There was a check plot variety every 6 plots within each column. That is the check variety was sown on rows 1,7,13,. . . ,67 of each column. This variety was numbered 526. A further 6 replicated commercially available varieties (numbered 527 to 532) were also randomly assigned to plots with between 3 to 5 plots of each. The aim of these trials is to identify and retain the top, say 20% of 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.

We consider an initial analysis with spatial correlation in one direction and fitting the variety effects (check, replicated and unreplicated lines) as random. We present three further spatial models for comparison. The ASReml input file is

```
!EPS !RENAME !ARG 1 2 3
Tullibigeal trial !DOPART $1
  linenum
  yield
  weed
  column 10
  row 67
  variety 532   # testlines 1:525, check lines 526:532
wheat.asd !SKIP 1
!PATH 1 # AR1 x I
y ~ mu weed mv !r idv(variety)
residual ar1v(row).id(col)
!PATH 2 # AR1 x AR1
y ~ mu weed mv !r variety
residual ar1v(row).ar1(col)
!PATH 3 # AR1 x AR1 + column trend
y ~ mu weed pol(column,-1) mv !r idv(variety)
residual ar1v(row).ar1(col)
!PATH 4 # AR1 x AR1 + Nugget + column trend
y ~ mu weed pol(column,-1) mv !r idv(variety) idv(units)
residual ar1(row).ar1(col)
predict var
```

The data fields represent the factors `variety, row` and `column`, a covariate `weed` and the plot yield (`yield`). There are four paths in the ASReml file. We begin with the one-dimensional spatial model, which assumes the variance model for the plot effects within columns is described by a first order autoregressive process. The abbreviated output file is

```
   1 LogL=-4280.75     S2= 0.12850E+06    666 df
   2 LogL=-4268.58     S2= 0.12139E+06    666 df
   3 LogL=-4255.89     S2= 0.10969E+06    666 df
   4 LogL=-4243.76     S2=  88040.        666 df
   5 LogL=-4240.59     S2=  84420.        666 df
```

```
  6 LogL=-4240.01     S2=  85617.        666 df
  7 LogL=-4239.91     S2=  86032.        666 df
  8 LogL=-4239.88     S2=  86189.        666 df
  9 LogL=-4239.88     S2=  86253.        666 df
 10 LogL=-4239.88     S2=  86280.        666 df

         - - - Results from analysis of yield - - -
 Akaike Information Criterion     8485.76 (assuming 3 parameters).
 Bayesian Information Criterion   8499.26

 Model_Term                               Gamma         Sigma   Sigma/SE   % C
 idv(variety)            IDV_V  532   0.959184       82758.6       8.98    0 P
 ar1(row).id(column)            670 effects
 Residual                SCA_V  670   1.000000       86280.2       9.12    0 P
 row                      AR_R    1   0.672052      0.672052      16.04    1 P

                                 Wald F statistics
     Source of Variation          NumDF      DenDF     F-inc            P-inc
   7 mu                               1       83.6   9799.20            <.001
   3 weed                             1      477.0    109.33            <.001
```

The iterative sequence converged, the REML estimate of the autoregressive parameter indicating substantial within column heterogeneity.

The abbreviated output from the two-dimensional AR1×AR1 spatial model is

```
  1 LogL=-4277.99     S2= 0.12850E+06    666 df
  2 LogL=-4266.14     S2= 0.12097E+06    666 df
  3 LogL=-4253.06     S2= 0.10778E+06    666 df
  4 LogL=-4238.72     S2=  83163.        666 df
  5 LogL=-4234.53     S2=  79867.        666 df
  6 LogL=-4233.78     S2=  82024.        666 df
  7 LogL=-4233.67     S2=  82724.        666 df
  8 LogL=-4233.65     S2=  82975.        666 df
  9 LogL=-4233.65     S2=  83065.        666 df
 10 LogL=-4233.65     S2=  83100.        666 df

         - - - Results from analysis of yield - - -
 Akaike Information Criterion     8475.29 (assuming 4 parameters).
 Bayesian Information Criterion   8493.30

 Model_Term                               Gamma         Sigma   Sigma/SE   % C
 idv(variety)            IDV_V  532    1.06038       88117.5       9.92    0 P
 ar1(row).ar1(column)           670 effects
 Residual                SCA_V  670   1.000000       83100.1       8.90    0 P
 row                      AR_R    1   0.685387      0.685387      16.65    0 P
 column                   AR_R    1   0.285909      0.285909       3.87    0 P

                                   Wald F statistics
     Source of Variation          NumDF      DenDF     F-inc            P-inc
   7 mu                               1       41.7   6248.66            <.001
   3 weed                             1      491.2     85.84            <.001
```

The change in REML log-likelihood is significant ($\chi^2_1 = 12.46, p < .001$) with the inclusion of the autoregressive parameter for columns. Figure 16.6 presents the sample variogram of the residuals for the AR1×AR1 model. There is an indication that a linear drift from column 1 to column 10 is present. We include a linear regression coefficient `pol(column,-1)` in the model to account for this. Note we use the '-1' option in the `pol` term to exclude the overall constant in the regression, as it is already fitted. The linear regression of column number on yield is significant ($t = -2.96$). The sample variogram (Figure 16.7) is more satisfactory, though interpretation of variograms is often difficult, particularly for unreplicated trials. This is an issue for further research.

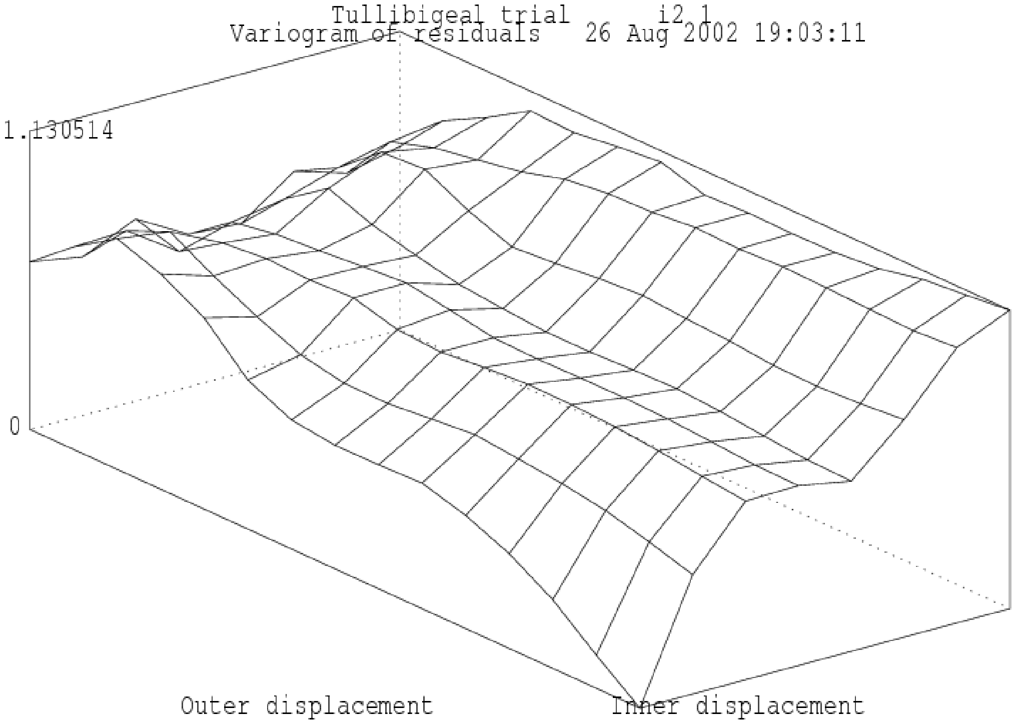

Figure 16.6: Sample variogram of the residuals from the AR1×AR1 model for the Tullibigeal data

The abbreviated output for this model and the final model in which a nugget effect has been included is

```
#AR1xAR1 + pol(column,-1)
   1 LogL=-4271.06     S2= 0.12731E+06    665 df
   2 LogL=-4259.03     S2= 0.11963E+06    665 df
   3 LogL=-4245.41     S2= 0.10556E+06    665 df
   4 LogL=-4229.98     S2=  78754.        665 df
   5 LogL=-4226.66     S2=  75970.        665 df
   6 LogL=-4226.29     S2=  77975.        665 df
   7 LogL=-4226.25     S2=  78313.        665 df
   8 LogL=-4226.25     S2=  78396.        665 df
   9 LogL=-4226.25     S2=  78419.        665 df
```

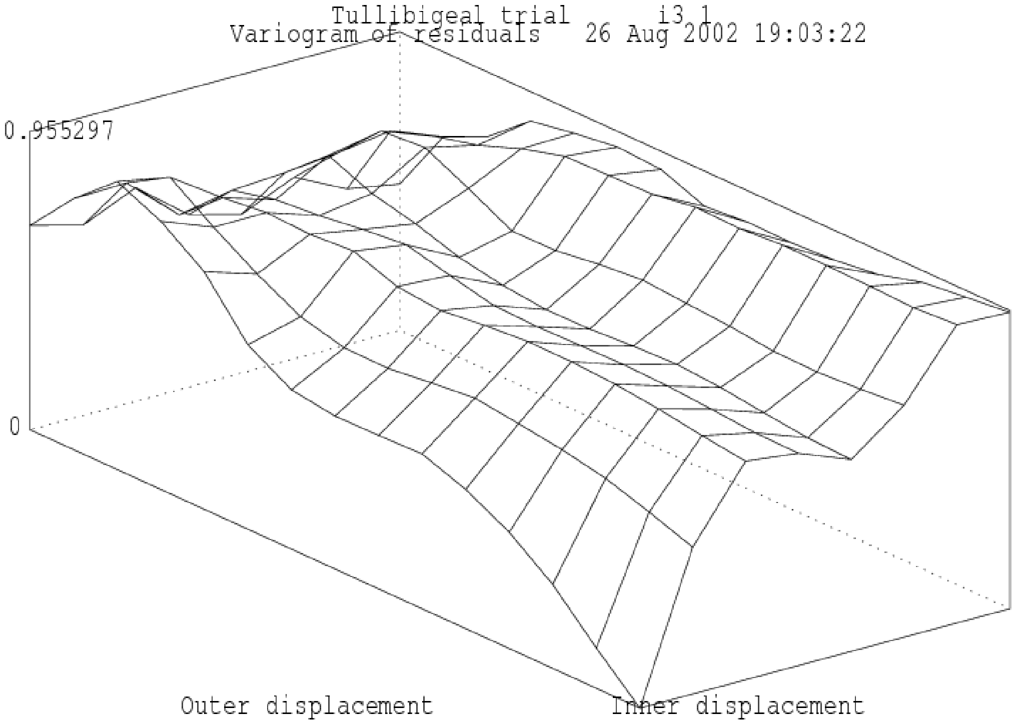

Figure 16.7: Sample variogram of the residuals from the AR1×AR1 + `pol(column,-1)` model for the Tullibigeal data

```
  10 LogL=-4226.25     S2=  78425.        665 df

         - - - Results from analysis of yield - - -
 Akaike Information Criterion     8460.50 (assuming 4 parameters).
 Bayesian Information Criterion   8478.50

 Model_Term                             Gamma         Sigma   Sigma/SE   % C
 idv(variety)            IDV_V  532   1.12313       88081.9       9.81   0 P
 ar1(row).ar1(column)               670  effects
 Residual                SCA_V  670   1.000000      78425.4       8.83   0 P
 row                      AR_R    1   0.665872     0.665872      15.37   0 P
 column                   AR_R    1   0.266047     0.266047       3.53   0 P

                                  Wald F statistics
    Source of Variation           NumDF     DenDF    F-inc            P-inc
   7 mu                               1      42.5  7149.90            <.001
   3 weed                             1     459.0    92.14            <.001
   8 pol(column,-1)                   1      62.1     7.61            0.008
#
#AR1xAR1 + units + pol(column,-1)
   1 LogL=-4272.85     S2= 0.11684E+06    665 df
   2 LogL=-4265.70     S2=  83872.        665 df    :   1 components restrained
   3 LogL=-4240.99     S2=  80942.        665 df
   4 LogL=-4227.44     S2=  53712.        665 df
   5 LogL=-4221.09     S2=  52201.        665 df
   6 LogL=-4220.94     S2=  54803.        665 df
   7 LogL=-4220.94     S2=  54935.        665 df
   8 LogL=-4220.94     S2=  54934.        665 df

          - - - Results from analysis of yield - - -
 Akaike Information Criterion     8451.88 (assuming 5 parameters).
 Bayesian Information Criterion   8474.37

 Model_Term                             Gamma         Sigma   Sigma/SE   % C
 idv(variety)            IDV_V  532   1.32827       72967.0       6.99   0 P
 idv(units)              IDV_V  670  0.562308       30889.9       3.78   0 P
 ar1(row).ar1(column)           670 effects
 Residual                SCA_V  670  1.000000       54934.0       5.15   0 P
 row                      AR_R    1  0.835396      0.835396      18.38   0 P
 column                   AR_R    1  0.375499      0.375499       3.25   0 P

                                   Wald F statistics
     Source of Variation           NumDF     DenDF     F-inc            P-inc
   7 mu                                1      13.6   4272.13            <.001
   3 weed                              1     470.3     86.31            <.001
   8 pol(column,-1)                    1      27.4      3.69            0.065
```

The increase in REML log-likelihood is significant. The predicted means for the varieties can be produced and printed in the `.pvs` file as

```
Ecode is E for Estimable, * for Not Estimable

 Warning: mv_estimates         is ignored for prediction
 Warning: units                is ignored for prediction


 ---- ---- ---- ---- ---- ----    1  ---- ---- ---- ---- ---- ----
 Predicted values of yield
 column is evaluated at       5.5000
 Model terms involving  weed  are predicted at the average:       0.4597

 variety         Predicted_Value Standard_Error Ecode
 1                     2916.6768       179.5421 E
 2                     2955.1002       179.0278 E
 3                     2869.7482       177.2955 E
 4                     2982.5846       178.9939 E
...
 522                   2777.5127       179.3317 E
 523                   2907.1301       179.7729 E
 524                   2776.0280       180.3853 E
 525                   2716.1221       181.8923 E
 526                   2381.9697        44.1852 E
 527                   2696.4092       133.8687 E
 528                   2723.5890       112.6784 E
 529                   2701.6306       104.2832 E
 530                   3006.8237       112.7234 E
 531                   3019.5559       112.6742 E
 532                   3064.3052       113.0868 E
 SED: Overall Standard Error of Difference    246.2
```

Note that the (replicated) check lines have lower SE than the (unreplicated) test lines. There will also be large diffeneces in SEDs. Rather than obtaining the large table of all SEDs, you could do the prediction in parts

```
predict var 1:525 column 5.5
predict var 526:532 column 5.5 !SED
```

to examine the matrix of pairwise prediction errors of variety differences.

16.8 Paired Case-Control study - Rice

This data is concerned with an experiment conducted to investigate the tolerance of rice varieties to attack by the larvae of bloodworms. The data have been kindly provided by Dr. Mark Stevens, Yanco Agricultural Institute. A full description of the experiment is given by Stevens *et al.* (1999). Bloodworms are a significant pest of rice in the Murray and Murrumbidgee irrigation areas where they can cause poor establishment and substantial yield loss.

The experiment commenced with the transplanting of rice seedlings into trays. Each tray contained 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. At the end of this time 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.

In the next three sections we present an exhaustive analysis of these data using equivalent univariate and multivariate techniques. It is convenient to use two data files one for each approach. The univariate data file consists of factors `pair`, `run`, `variety`, `tmt`, `unit` and variate `rootwt`. The factor `unit` labels the individual trays, `pair` labels pairs of trays (to which varieties are allocated) and `tmt` is the two level bloodworm treatment factor (control/treated). The multivariate data file consists of factors `variety` and `run` and variates for root weight of both the control and exposed treatments (labelled `yc` and `ye` respectively).

Preliminary analyses indicated variance heterogeneity so that subsequent analyses were conducted on the square root scale. Figure 16.8 presents a plot of the treated and the control root area (on the square root scale) for each variety. 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 thence identify the most tolerant varieties. The definition of tolerance should allow for the fact that varieties differ in their inherent seedling vigour (Figure 16.8). The original approach of the scientist 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 approach 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.

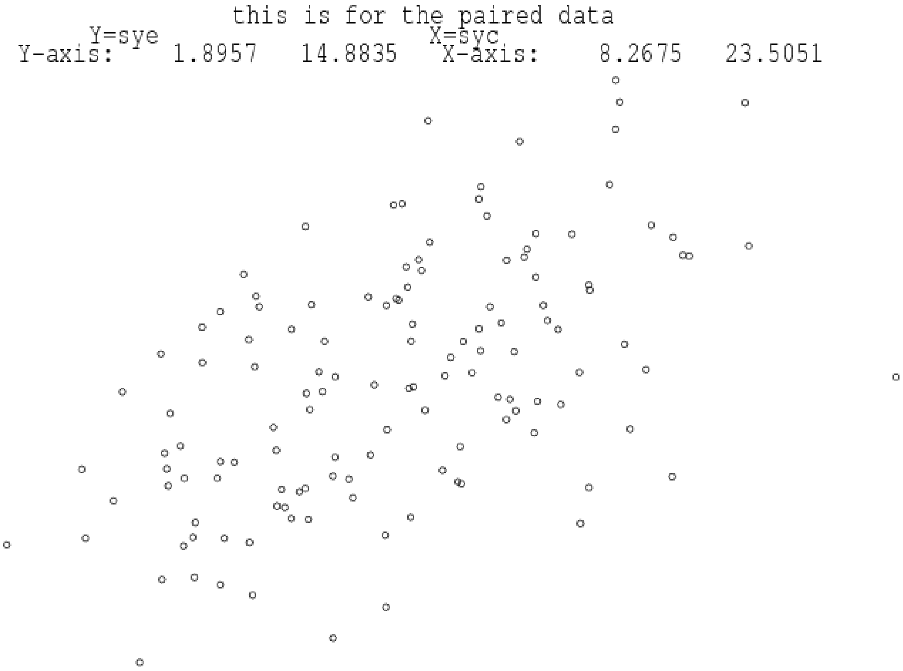

Figure 16.8: Rice bloodworm data: Plot of square root of root weight for treated versus control

16.8.1 Standard analysis

The allocation of bloodworm treatments within varieties and varieties within runs defines a nested block structure of the form

```
run/variety/tmt = run + run.variety + run.variety.tmt
              ( = run + pair + pair.tmt )
              ( = run + run.variety + units )
```

There is an additional blocking term, however, due to the fact that the bloodworms within a run are derived from the same batch of larvae whereas between runs the bloodworms come from different sources. This defines a block structure of the form

```
run/tmt/variety = run + run.tmt + run.tmt.variety
              ( = run + run.tmt + pair.tmt )
```

Combining the two provides the full block structure for the design, namely

```
  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. Thus to minimise selection bias the variety main effects and thence the `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 BLUPs of `tmt.variety` effects are shrunk to the correct mean (treatment means rather than an overall mean).

The model for the data is given by

$$\boldsymbol{y} = \boldsymbol{X\tau} + \boldsymbol{Z}_1\boldsymbol{u}_1 + \boldsymbol{Z}_2\boldsymbol{u}_2 + \boldsymbol{Z}_3\boldsymbol{u}_3 + \boldsymbol{Z}_4\boldsymbol{u}_4 + \boldsymbol{Z}_5\boldsymbol{u}_5 + \boldsymbol{e} \tag{16.7}$$

where $\boldsymbol{y}$ is a vector of length $n = 264$ containing the sqrt(rootwt) values, $\boldsymbol{\tau}$ corresponds to a constant term and the fixed treatment contrast and $\boldsymbol{u}_1 \ldots \boldsymbol{u}_5$ correspond to random variety, treatment by variety, run, treatment by run and variety by run effects. The random effects and error are assumed to be independent Gaussian variables with zero means and variance structures $\text{var}\,(\boldsymbol{u}_i) = \sigma_i^2 \boldsymbol{I}_{b_i}$ (where b_i is the length of $\boldsymbol{u}_i$, $i = 1 \ldots 5$) and $\text{var}\,(\boldsymbol{e}) = \sigma^2 \boldsymbol{I}_n$.

The ASReml code for this analysis is

```
Bloodworm data Dr M Stevens
 pair 132
 rootwt
 run 66
 tmt 2 !A
 id
 variety 44 !A
rice.asd !skip 1 !DOPATH 1
!PATH 1
sqrt(rootwt) ~ mu tmt !r idv(variety) idv(variety.tmt) idv(run) ,
idv(pair) idv(run.tmt)
residual idv(units)
!PATH 2
sqrt(rootwt) ~ mu tmt !r idv(variety) diag(tmt).id(variety) idv(run),
 idv(pair) diag(tmt).id(run) idv(uni(tmt,2))
residual idv(units)
```

The two paths in the input file define the two univariate analyses we will conduct. We consider the results from the analysis defined in `PATH 1` first. A portion of the output file is

```
   5 LogL=-345.306     S2=  1.3216        262 df
   6 LogL=-345.267     S2=  1.3155        262 df
   7 LogL=-345.264     S2=  1.3149        262 df
   8 LogL=-345.263     S2=  1.3149        262 df

          - - - Results from analysis of sqrt(rootwt) - - -
 Akaike Information Criterion      702.53 (assuming 6 parameters).
 Bayesian Information Criterion    723.94

          Approximate stratum variance decomposition
 Stratum     Degrees-Freedom   Variance      Component Coefficients
 idv(variety)          44.40    26.0156        7.3      3.0      3.6      2.0      1.5      1.0
 idv(run)              45.17    7.41702        0.0      3.5     -0.0      2.0      1.7      1.0
 idv(variety.tmt)      39.53    2.99833        0.0      0.0      2.7     -0.0      0.2      1.0
 idv(pair)             41.43    3.26838        0.0      0.0      0.0      2.0     -0.0      1.0
 idv(run.tmt)          52.38    5.12369        0.0      0.0      0.0      0.0      2.2      1.0
 Residual Variance     39.09    1.31486        0.0      0.0      0.0      0.0      0.0      1.0

 Model_Term                              Gamma         Sigma   Sigma/SE   % C
 idv(variety)            IDV_V    44    1.80947       2.37920       3.01   0 P
 idv(run)                IDV_V    66   0.244243      0.321145       0.59   0 P
 idv(variety.tmt)        IDV_V    88   0.374220      0.492047       1.78   0 P
 idv(pair)               IDV_V   132   0.742328      0.976056       2.51   0 P
 idv(run.tmt)            IDV_V   132    1.32973       1.74841       3.65   0 P
 idv(units)                      264 effects
 Residual                SCA_V   264   1.000000       1.31486       4.42   0 P

                                               Wald F statistics
     Source of Variation                       NumDF     DenDF     F-inc             P-inc
   7 mu                                            1      53.6   1484.96             <.001
   4 tmt                                           1      60.4    469.35             <.001
```

Table 16.8: Estimated variance components from univariate analyses of bloodworm data. (a) Model with homogeneous variance for all terms and (b) Model with heterogeneous variance for interactions involving tmt

	(a)	(b)	
source		control	treated
variety	2.378	2.334	
tmt.variety	0.492	1.505	-0.372
run	0.321	0.319	
tmt.run	1.748	1.388	2.223
variety.run (pair)	0.976	0.987	
tmt.pair	1.315	1.156	1.359
REML log-likelihood	-345.256	-343.22	

The estimated variance components from this analysis are given in column (a) of table 16.8. The variance component for the `variety` main effects is large. There is evidence of `tmt.variety` 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` variance for each level of `tmt` so that instead of assuming $\text{var}\,(\boldsymbol{u}_2) = \sigma_2^2 \boldsymbol{I}_{88}$ we assume

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

where σ_{2c}^2 and σ_{2t}^2 are the `tmt.variety` interaction variances for control and treated respectively. This model can be achieved using a diagonal variance structure for the treatment part of the interaction. We also fit a separate `run` variance for each level of `tmt` and heterogeneity at the residual level, by including the `uni(tmt,2)` 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. Had we mistakenly specified level 1 then ASReml would have estimated a negative component by setting the `!GU` option for this term. The portion of the ASReml output for this analysis is

```
  5 LogL=-343.759     S2=  1.2242        262 df    :    1 components restrained
  6 LogL=-343.577     S2=  1.1738        262 df    :    1 components restrained
  7 LogL=-343.543     S2=  1.1559        262 df
  8 LogL=-343.535     S2=  1.1469        262 df
  9 LogL=-343.535     S2=  1.1451        262 df

          - - - Results from analysis of sqrt(rootwt) - - -
 Akaike Information Criterion      705.07 (assuming 9 parameters).
 Bayesian Information Criterion    737.18
 Model_Term                                  Gamma         Sigma   Sigma/SE   % C
 idv(variety)            IDV_V    44    1.89172       2.16613       2.99    0 P
 idv(run)                IDV_V    66   0.296929      0.340000       0.62    0 P
 idv(pair)               IDV_V   132   0.871770      0.998227       2.64    0 P
 idv(uni(tmt,2))         IDV_V   264   0.144454      0.165408       0.27    0 P
 idv(units)                      264  effects
 Residual                SCA_V   264   1.000000       1.14506       2.79    0 P
 diag(tmt).id(variety)            88  effects
 tmt                    DIAG_V     1    1.09032       1.24848       2.21    0 P
 tmt                    DIAG_V     2  0.148952E-05  0.170558E-05    2.79    0 B
 diag(tmt).id(run)               132  effects
 tmt                    DIAG_V     1    1.25736       1.43975       2.25    0 P
 tmt                    DIAG_V     2    1.86671       2.13750       2.97    0 P
 Warning: Code B - fixed at a boundary (!GP)      F - fixed by user
               ? - liable to change from P to B   P - positive definite
               C - Constrained by user (!VCC)     U - unbounded
               S - Singular Information matrix
 S means there is no information in the data for this parameter.
 Very small components with Comp/SE ratios of zero sometimes indicate poor
           scaling.  Consider rescaling the design matrix in such cases.

                                   Wald F statistics
     Source of Variation           NumDF              F-inc
   7 mu                                1            1405.14
   4 tmt                               1             441.72
```

The estimated variance components from this analysis are given in column (b) of table 16.8. There is no significant variance heterogeneity at the residual or `tmt.run` level. This indicates that the square root transformation of the data has successfully stabilised the error variance. There is, however, significant variance heterogeneity for `tmt.variety` 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 `tmt.variety` interaction variance for the treated group is negative. The negative component is meaningful (and in fact necessary and obtained by use of the `!GU` option) in this context since it should be considered as part of the variance structure for the combined variety main effects and treatment by variety interactions. That is,

$$\text{var}\left(\mathbf{1}_2 \otimes \boldsymbol{u}_1 + \boldsymbol{u}_2\right) = \begin{bmatrix} \sigma_1^2 + \sigma_{2c}^2 & \sigma_1^2 \\ \sigma_1^2 & \sigma_1^2 + \sigma_{2t}^2 \end{bmatrix} \otimes \boldsymbol{I}_{44} \tag{16.8}$$

Using the estimates from table 16.8 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 $2.33/\sqrt{3.84 * 1.96} = 0.85$ which is strong, supporting earlier indications of the dependence between the treated and control root area (Figure 16.8).

Table 16.9: Equivalence of random effects in bivariate and univariate analyses

effects	bivariate (model 16.10)	univariate (model 16.7)
trait.variety	$\boldsymbol{u}_v$	$\mathbf{1}_2 \otimes \boldsymbol{u}_1 + \boldsymbol{u}_2$
trait.run	$\boldsymbol{u}_r$	$\mathbf{1}_2 \otimes \boldsymbol{u}_3 + \boldsymbol{u}_4$
trait.pair	$\boldsymbol{e}^*$	$\mathbf{1}_2 \otimes \boldsymbol{u}_5 + \boldsymbol{e}$

16.8.2 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 sqrt(rootwt) for control and treated. The model for each trait is given by

$$\boldsymbol{y}_j = \boldsymbol{X}\boldsymbol{\tau}_j + \boldsymbol{Z}_v\boldsymbol{u}_{v_j} + \boldsymbol{Z}_r\boldsymbol{u}_{r_j} + \boldsymbol{e}_j \quad (j = c, t) \tag{16.9}$$

where $\boldsymbol{y}_j$ is a vector of length $n = 132$ containing the sqrtroot values for variate j ($j = c$ for control and $j = t$ for treated), $\boldsymbol{\tau}_j$ corresponds to a constant term and $\boldsymbol{u}_{v_j}$ and $\boldsymbol{u}_{r_j}$ correspond to random variety and run effects. The design matrices are the same for both traits. The random effects and error are assumed to be independent Gaussian variables with zero means and variance structures var $(\boldsymbol{u}_{v_j}) = \sigma^2_{v_j}\boldsymbol{I}_{44}$, var $(\boldsymbol{u}_{r_j}) = \sigma^2_{r_j}\boldsymbol{I}_{66}$ and var $(\boldsymbol{e}_j) = \sigma^2_j\boldsymbol{I}_{132}$. The bivariate model can be written as a direct extension of (16.9), namely

$$\boldsymbol{y} = (\boldsymbol{I}_2 \otimes \boldsymbol{X})\,\boldsymbol{\tau} + (\boldsymbol{I}_2 \otimes \boldsymbol{Z}_v)\,\boldsymbol{u}_v + (\boldsymbol{I}_2 \otimes \boldsymbol{Z}_r)\,\boldsymbol{u}_r + \boldsymbol{e}^* \tag{16.10}$$

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 (16.7). The variety effects for each trait ($\boldsymbol{u}_v$ in the bivariate model) are partitioned in (16.7) into variety main effects and `tmt.variety` interactions so that $\boldsymbol{u}_v = \mathbf{1}_2 \otimes \boldsymbol{u}_1 + \boldsymbol{u}_2$. There is a similar partitioning for the run effects and the errors (see table 16.9).

In addition to the assumptions in the models for individual traits (16.9) 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

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

This unstructured form for `trait.variety` in the bivariate analysis is equivalent to the `variety` main effect plus heterogeneous `tmt.variety` interaction variance structure (16.8) in the univariate analysis. Similarly the unstructured form for `trait.run` is equivalent to the `run` main effect plus heterogeneous `tmt.run` interaction variance structure. The unstructured form for the errors (`trait.pair`) in the bivariate analysis is equivalent to the `pair` plus heterogeneous error (`tmt.pair`) variance in the univariate analysis. This bivariate analysis is achieved in ASReml as follows, noting that the tmt factor here is equivalent to traits.

```
this is for the paired data
 id
 pair 132
 run 66
 variety 44 !A
 yc ye
ricem.asd !skip 1 !X syc !Y sye
sqrt(yc) sqrt(ye) ~ Trait !r us(Trait).id(variety) us(Trait).id(run)
residual id(units).us(Trait)
predict variety
```

A portion of the output from this analysis is

```
   8 LogL=-343.220     S2= 1.00000        262 df
   9 LogL=-343.220     S2= 1.00000        262 df

         - - - Results from analysis of sqrt(yc) sqrt(ye) - - -
 Akaike Information Criterion      704.44 (assuming 9 parameters).
 Bayesian Information Criterion    736.56

 Model_Term                             Sigma         Sigma   Sigma/SE   % C
 id(units).us(Trait)            264 effects
 Trait                   US_V  1  1    2.14370       2.14370       4.44   0 P
 Trait                   US_C  2  1  0.987342      0.987342       2.59   0 P
 Trait                   US_V  2  2    2.34744       2.34744       4.62   0 P
 us(Trait).id(variety)           88 effects
 Trait                   US_V  1  1    3.83911       3.83911       3.47   0 P
 Trait                   US_C  2  1    2.33352       2.33352       3.01   0 P
 Trait                   US_V  2  2    1.96136       1.96136       2.69   0 P
 us(Trait).id(run)              132 effects
 Trait                   US_V  1  1    1.70810       1.70810       2.61   0 P
 Trait                   US_C  2  1  0.319444      0.319444       0.59   0 P
 Trait                   US_V  2  2    2.54360       2.54360       3.20   0 P
 Covariance/Variance/Correlation Matrix US Residual
   2.144      0.4401
  0.9873       2.347
 Covariance/Variance/Correlation Matrix US us(Trait).id(variety
   3.839      0.8504
   2.334       1.961
 Covariance/Variance/Correlation Matrix US us(Trait).id(run)
   1.708      0.1533
  0.3194       2.544
```

The resultant REML log-likelihood is identical to that of the heterogeneous univariate analysis (column (b) of table 16.8). The estimated variance parameters are given in Table 16.10.

The predicted variety means in the .pvs file are used in the following section on interpretation of results. A portion of the file is presented below. There is a wide range in SED reflecting the imbalance of the variety concurrence within runs.

```
Assuming Power transformation was (Y+0.000)^0.500
The ignored set: run

variety         Trait                 Power_value  Stand_Error Ecode Retransformed_value  approx_SE
AliCombo        sqrt(yc)                14.9531         0.9181 E          223.5962         27.4568
AliCombo        sqrt(ye)                 7.9941         0.7992 E           63.9050         12.7784
Bluebelle       sqrt(yc)                13.1036         0.9310 E          171.7046         24.3987
Bluebelle       sqrt(ye)                 6.6302         0.8062 E           43.9598         10.6903
C22             sqrt(yc)                16.6676         0.9181 E          277.8096         30.6050
C22             sqrt(ye)                 8.9541         0.7992 E           80.1756         14.3130
. . . . . . . .
YRK1            sqrt(yc)                15.1857         0.9549 E          230.6068         29.0018
YRK1            sqrt(ye)                 8.3355         0.8190 E           69.4806         13.6531
YRK3            sqrt(yc)                13.3058         0.9549 E          177.0431         25.4114
YRK3            sqrt(ye)                 8.1134         0.8190 E           65.8265         13.2892
```

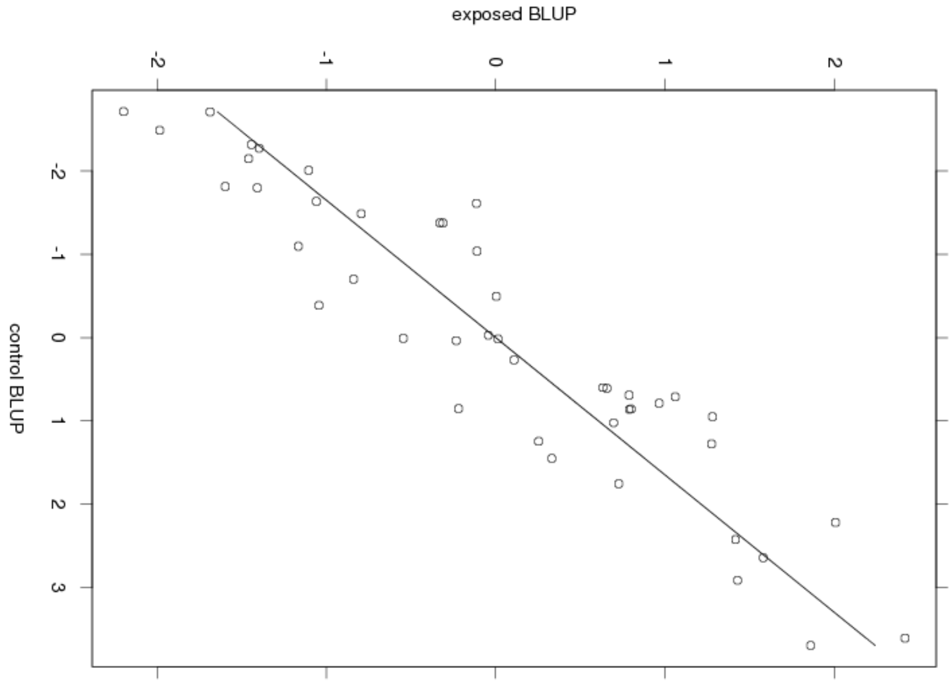

Figure 16.9: BLUPs for treated for each variety plotted against BLUPs for control

Table 16.10: Estimated variance parameters from bivariate analysis of bloodworm data

source	control variance	treated variance	covariance
us(trait).variety	3.84	1.96	2.33
us(trait).run	1.71	2.54	0.32
us(trait).pair	2.14	2.35	0.99

16.8.3 Interpretation of results

Recall that the researcher is interested in varietal tolerance to bloodworms. This could be defined in various ways. One option is to consider the regression implicit in the variance structure for the trait by variety effects. The variance structure can arise from a regression of treated variety effects on control effects, namely

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

where the slope $\beta = \sigma_{v_{ct}}/\sigma^2_{v_c}$. Tolerance can be defined in terms of the deviations from regression, $\boldsymbol{\epsilon}$. Varieties with large positive deviations have greatest tolerance to bloodworms. Note that this is similar to the researcher's original intentions except that the regression has been conducted at the genotypic rather than the phenotypic level. In Figure 16.9 the BLUPs for treated have been plotted against the BLUPs for control for each variety and the fitted regression line (slope = 0.61) has been drawn. Varieties with large positive deviations from the regression line include YRK3, Calrose, HR19 and WC1403.

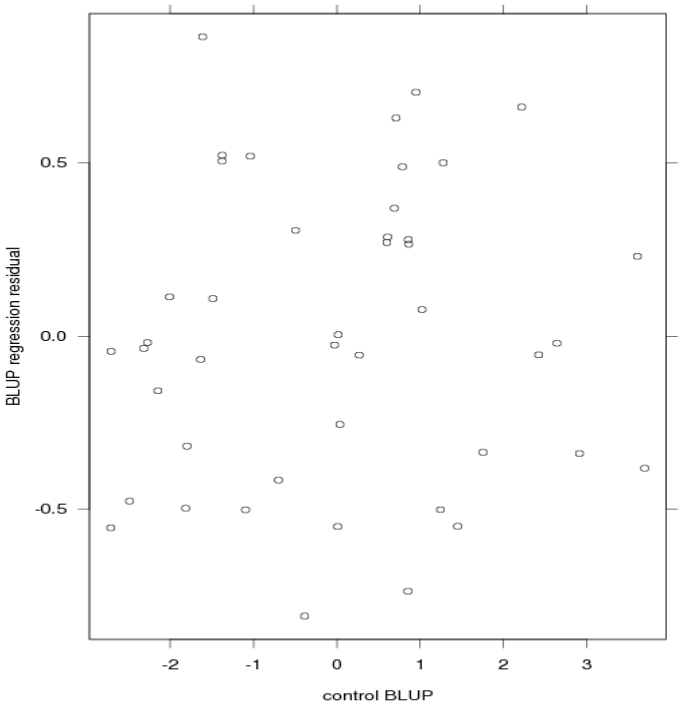

Figure 16.10: Estimated deviations from regression of treated on control for each variety plotted against estimate for control

An alternative definition of tolerance is the simple difference between treated and control 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

$$\text{cov}(\boldsymbol{\epsilon})\,\boldsymbol{u}'_{v_c} \quad = \quad \text{cov}(\boldsymbol{u}_{v_t} - \beta \boldsymbol{u}_{v_c})\,\boldsymbol{u}'_{v_c}$$

$$
\begin{aligned}
&= \left(\sigma_{v_{ct}} - \frac{\sigma_{v_{ct}}}{\sigma^2_{v_c}}\sigma^2_{v_c}\right)\boldsymbol{I}_{44} \\
&= \mathbf{0}
\end{aligned}
$$

whereas

$$
\begin{aligned}
\text{cov}\,(\boldsymbol{\delta})\,\boldsymbol{u}'_{v_c} &= \text{cov}\,(\boldsymbol{u}_{v_c} - \boldsymbol{u}_{v_t})\,\boldsymbol{u}'_{v_c} \\
&= \left(\sigma^2_{v_c} - \sigma_{v_{ct}}\right)\boldsymbol{I}_{44}
\end{aligned}
$$

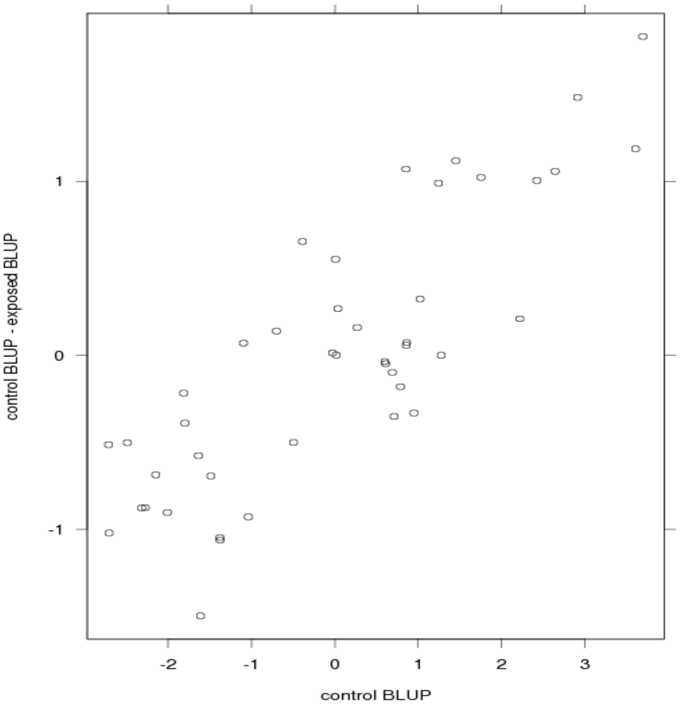

Figure 16.11: Estimated difference between control and treated for each variety plotted against estimate for control

The independence of $\boldsymbol{\epsilon}$ and $\boldsymbol{u}_{v_c}$ and dependence between $\boldsymbol{\delta}$ and $\boldsymbol{u}_{v_c}$ is clearly illustrated in Figures 16.10 and 16.11. In this example the two measures have provided very different rankings of the varieties. The choice of tolerance measure depends on the aim of the experiment. In this experiment the aim was to identify tolerance which is independent of inherent vigour so the deviations from regression measure is preferred.

16.9 Balanced longitudinal data - Random coefficients and cubic smoothing splines - Oranges

We now 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 was originally based on the mixed model formulation presented by Verbyla *et al.* (1999). More recently the technology has been enhanced 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 we use was originally reported by Draper and Smith (1998, ex24N, p559) and has recently been reanalysed by Pinheiro and Bates (2000, p338). The data are displayed in Figure 16.12 and are the trunk circumferences (in millimetres) of each of 5 trees taken at 7 times. All trees were measured at the same time so that the data are balanced. The aim of the study is unclear, though, both previous analyses involved modelling the overall 'growth' curve, accounting for the obvious variation in both level and shape between trees. Pinheiro and Bates (2000) used a nonlinear mixed effects modelling approach, in which they modelled the growth curves by a three parameter logistic function of age, given by

$$y = \frac{\phi_1}{1 + \exp\left[-(x - \phi_2)/\phi_3\right]} \tag{16.11}$$

where y is the trunk circumference, x is the tree age in days since December 31 1968, ϕ_1 is the asymptotic height, ϕ_2 is the inflection point or the time at which the tree reaches $0.5\phi_1$, ϕ_3 is the time elapsed between trees reaching half and about $3/4$ of ϕ_1.

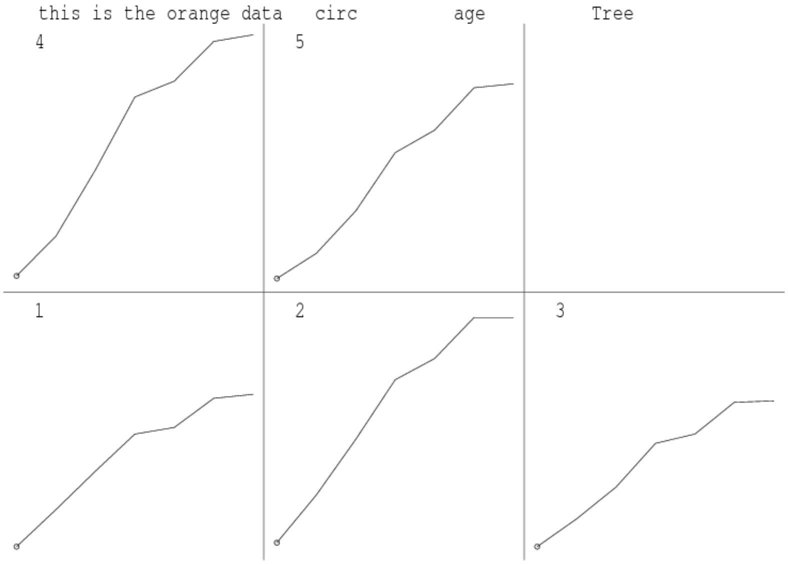

Figure 16.12: Trellis plot of trunk circumference for each tree

The datafile consists of 5 columns viz, `Tree`, a factor with 5 levels, `age,` tree age in days since 31st December 1968, `circ` the trunk circumference and `season`. The last column `season` was added after noting that tree age spans several years and if converted to day of year, measurements were taken in either `Spring` (April/May) or `Autumn` (September/October).

First we demonstrate the fitting of a cubic spline in ASReml by restricting the dataset to tree 1 only. The model includes the intercept and linear regression of trunk circumference on *age* and an additional random term `spl(age,7)` which instructs ASReml to include a random term with a special design matrix with $7-2=5$ columns which relate to the vector, $\boldsymbol{\delta}$ whose elements $\delta_i, i = 2, \ldots, 6$ are the second differentials of the cubic spline at the knot points. The second differentials of a natural cubic spline are zero at the first and last knot points (Green and Silverman, 1994). The ASReml job is

```
this is the orange data, for tree 1
 seq     # record number is not used
 Tree 5
 age       # 118  484  664 1004 1231 1372 1582
 circ
 season !L Spring Autumn
orange.asd !skip 1 !filter 2 !select 1
!SPLINE spl(age,7) 118  484  664 1004 1231 1372 1582
!PVAL age 150 200:1500
circ ~ mu age !r idv(spl(age,7))
residual idv(units)
predict age
```

Note that the data for tree 1 has been selected by use of the `!filter` and `!select` qualifiers. Also note the use of `!PVAL` so that the spline curve is properly predicted at the additional nominated points. These additional data points are required for ASReml to form the design matrix to properly interpolate the cubic smoothing spline between knot points in the prediction process. Since the spline knot points are specifically nominated in the `!SPLINE` line, these extra points have no effect on the analysis run time. The `!SPLINE` line does not modify the analysis in this example since it simply nominates the 7 ages in the data file. The same analysis would result if the `!SPLINE` line was omitted and `spl(age,7)` in the model was replaced with `spl(age)`. An extract of the output file is

```
   1 LogL=-20.9043    S2=  48.470          5 df   0.1000E+00
   2 LogL=-20.9013    S2=  49.152          5 df   0.9102E-01
   3 LogL=-20.8998    S2=  49.892          5 df   0.8221E-01
   4 LogL=-20.8996    S2=  50.273          5 df   0.7802E-01
 Final parameter values                        0.7892E-01

            - - - Results from analysis of circ - - -
 Akaike Information Criterion       45.80 (assuming 2 parameters).
 Bayesian Information Criterion     45.02

          Approximate stratum variance decomposition
 Stratum     Degrees-Freedom   Variance      Component Coefficients
 idv(spl(age,7))        1.49    98.4896        12.2     1.0
 Residual Variance      3.51    50.2726         0.0     1.0

 Model_Term                             Gamma         Sigma   Sigma/SE   % C
 idv(spl(age,7))         IDV_V    5  0.789210E-01   3.96756       0.40   0 P
```

```
Residual                SCA_V    7  1.000000       50.2726       1.32   0 P
Notice: The DenDF values are calculated ignoring fixed/boundary/singular
            variance parameters using algebraic derivatives.

                       Estimate       Standard Error    T-value     T-prev
  3 age
                     1   0.814772E-01   0.552336E-02     14.75
  7 mu
                     1    24.4378        5.75429          4.25
  6 spl(age,7)                              5 effects fitted
Finished: 19 Aug 2005 10:08:11.980   LogL Converged
```

The REML estimate of the smoothing constant indicates that there is some nonlinearity. The fitted cubic smoothing spline is presented in Figure 16.13. The fitted values were obtained from the .pvs file. The four points below the line were the spring measurements.

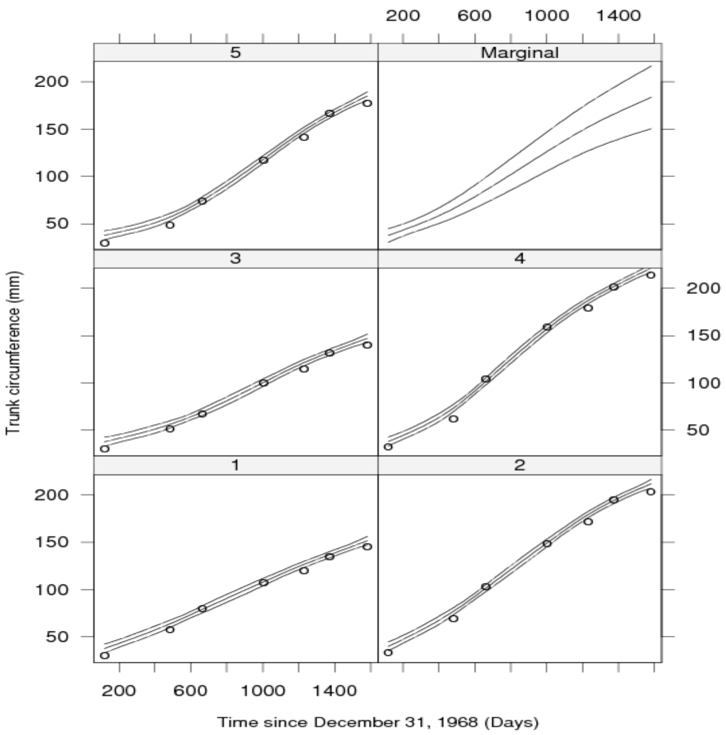

Figure 16.13: Fitted cubic smoothing spline for tree 1

We now consider the analysis of the full dataset. Following Verbyla *et al.* (1999) we consider the analysis of variance decomposition (see Table 16.11) which models the overall and individual curves.

An overall spline is fitted as well as tree deviation splines. We note however, that the intercept and slope for the tree deviation splines are assumed to be random effects. This is consistent with Verbyla *et al.* (1999). In this sense the tree deviation splines play a role in modelling the conditional curves for each tree and variance modelling. The intercept and

Table 16.11: Orange data: AOV decomposition

stratum	decomposition	type	df or ne
constant	1	F	1
age			
	age	F	1
	spl(age,7)	R	5
	fac(age)	R	7
tree			
	tree	RC	5
age.tree			
	x.tree	RC	5
	spl(age,7).tree	R	25
error		R	

slope for each tree are included as random coefficients (denoted by RC in Table 16.11). Thus, if $\boldsymbol{U}^{5\times 2}$ is the matrix of intercepts (column 1) and slopes (column 2) for each tree, then we assume that

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

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 in ASReml by inclusion of `fac(age)` as a random term.

Table 16.12: Sequence of models fitted to the Orange data

	model					
term	1	2	3	4	5	6
tree	y	y	y	y	y	y
age.tree	y	y	y	y	y	y
(covariance)	n	n	n	n	n	y
spl(age,7)	y	y	y	y	n	y
tree.spl(age,7)	y	y	y	n	y	y
fac(age)	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

An extract of the ASReml input file is

```
circ ~ mu age !r str(Tree age.Tree  us(2 !INIT 4.6 .00001 .000094).id(Tree)) idv(spl(age,7)),
idv(spl(age,7).Tree) idv(fac(age))
predict age Tree !IGNORE fac(age)
```

We stress the importance of model building in these settings, where we generally commence with relatively simple variance models and update to more complex variance models if appropriate. Table 16.12 presents the sequence of fitted models we have used. Note that the REML log-likelihoods for models 1 and 2 are comparable and likewise for models 3 to 6. The REML log-likelihoods are not comparable between these groups due to the inclusion of the fixed `season` term in the second set of models.

We begin by modelling the variance matrix for the intercept and slope for each tree, $\mathbf{\Sigma}$, as a diagonal matrix as there is no point including a covariance component between the intercept and slope if the variance component(s) for one (or both) is zero. Model 1 also does not include a non-smooth component at the overall level (that is, `fac(age)`). Abbreviated output is shown below.

```
  6 LogL=-97.8517     S2=  7.2838         33 df
  7 LogL=-97.7837     S2=  6.6673         33 df
  8 LogL=-97.7792     S2=  6.4634         33 df
  9 LogL=-97.7788     S2=  6.3911         33 df
 10 LogL=-97.7788     S2=  6.3615         33 df

          - - - Results from analysis of circ - - -
 Akaike Information Criterion      205.56 (assuming 5 parameters).
 Bayesian Information Criterion    213.04

 Model_Term                                 Gamma         Sigma   Sigma/SE   % C
 idv(spl(age,7))         IDV_V    5   100.466       639.116       1.55    0 P
 Residual                SCA_V   35  1.000000       6.36154       1.74    0 P
 idv(Tree)                ID_V    1   4.78778       30.4577       1.24    0 P
 idv(Tree.age)            ID_V    1  0.939009E-04  0.597354E-03   1.41    0 P
 idv(spl(age,7).Tree)     ID_V    1   1.11619       7.10070       1.44    0 P

                                 Wald F statistics
     Source of Variation         NumDF     DenDF    F-inc             P-inc
   9 mu                              1       4.0    47.05             0.002
   3 age                             1       4.0    95.00             <.001
```

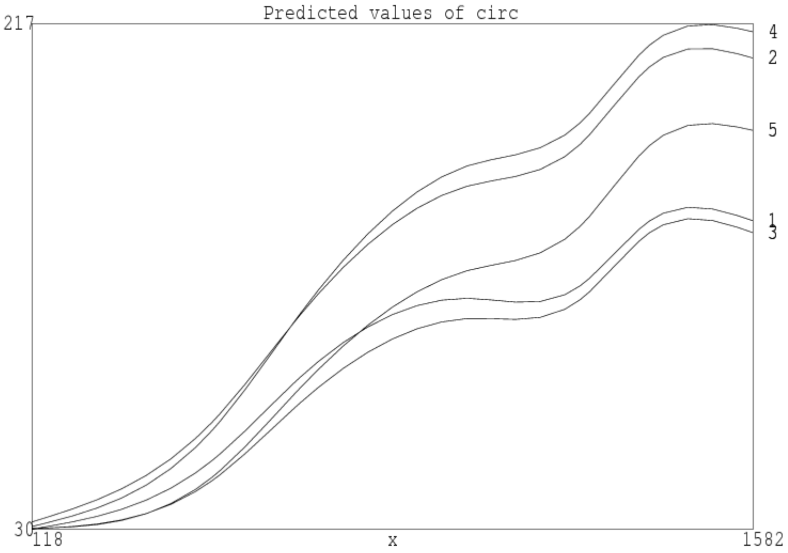

Figure 16.14: Plot of fitted cubic smoothing spline for model 1

A quick look suggests this is fine until we look at the predicted curves in Figure 16.14. The fit is unacceptable because the spline has picked up too much curvature, and suggests that there may be systematic non-smooth variation at the overall level. This can be formally examined by including the `fac(age)` term as a random effect. This increased the log-likelihood 3.71 ($P < 0.05$) with the `spl(age,7)` smoothing constants heading to the boundary. There is a possible explanation in the `season` factor. When this is added (Model 3) it has an F ratio of 107.5 ($P < 0.01$) while the `fac(age)` term goes to the boundry. Notice that the inclusion of the fixed term `season` in models 3 to 6 means that comparisons with models 1 and 2 on the basis of the log-likelihood are not valid. 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 ($P < 0.05$). Lastly we add the covariance parameter between the intercept and slope for each tree in model 6. This ensures that the covariance model will be translation invariant. A portion of the output file for model 6 is

```
   6 LogL=-87.5371      S2=   5.9488         32 df
   7 LogL=-87.4342      S2=   5.6885         32 df
   8 LogL=-87.4291      S2=   5.6434         32 df
   9 LogL=-87.4291      S2=   5.6412         32 df
```

```
         - - - Results from analysis of circ - - -
 Akaike Information Criterion      186.86 (assuming 6 parameters).
 Bayesian Information Criterion    195.65

 Model_Term                              Gamma         Sigma   Sigma/SE   % C
 idv(spl(age,7))         IDV_V    5    2.17100       12.2471       1.09   0 P
 Residual                SCA_V   35   1.000000       5.64123       1.72   0 P
 us(2).id(Tree)                  10 effects
 2                       US_V  1  1    5.61715       31.6877       1.26   0 P
 2                       US_C  2  1 -0.124098E-01 -0.700063E-01  -0.85   0 P
 2                       US_V  2  2  0.108290E-03  0.610886E-03   1.41   0 P
 idv(spl(age,7).Tree)     ID_V    1    1.38313       7.80258       1.48   0 P
 Covariance/Variance/Correlation Matrix US Tree
   31.69     -0.5032
 -0.7001E-01  0.6109E-03


                                  Wald F statistics
     Source of Variation          NumDF     DenDF    F-inc             P-inc
   9 mu                               1       2.4   169.87             0.006
   3 age                              1       2.4    92.78             0.011
   5 Season                           1       8.8   108.49             <.001
```

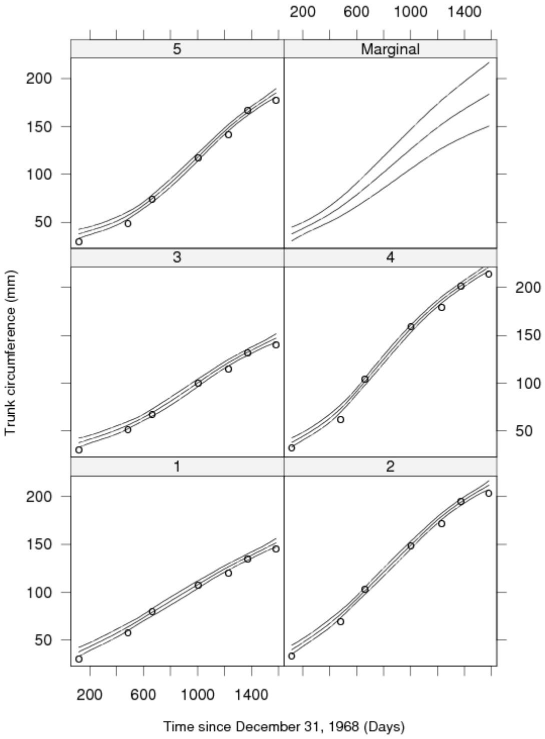

Figure 16.15: Trellis plot of trunk circumference for each tree at sample dates (adjusted for *season* effects), with fitted profiles across time and confidence intervals

Figure 16.15 presents the predicted growth over time for individual trees and a marginal prediction for trees with approximate confidence intervals (2±× standard 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. Figure 16.16 presents a plot of the residuals from the nonlinear model fitted on p340 of Pinheiro and Bates (2000). The distinct pattern in the residuals, which is the same for all trees is taken up in our analysis by the season term.

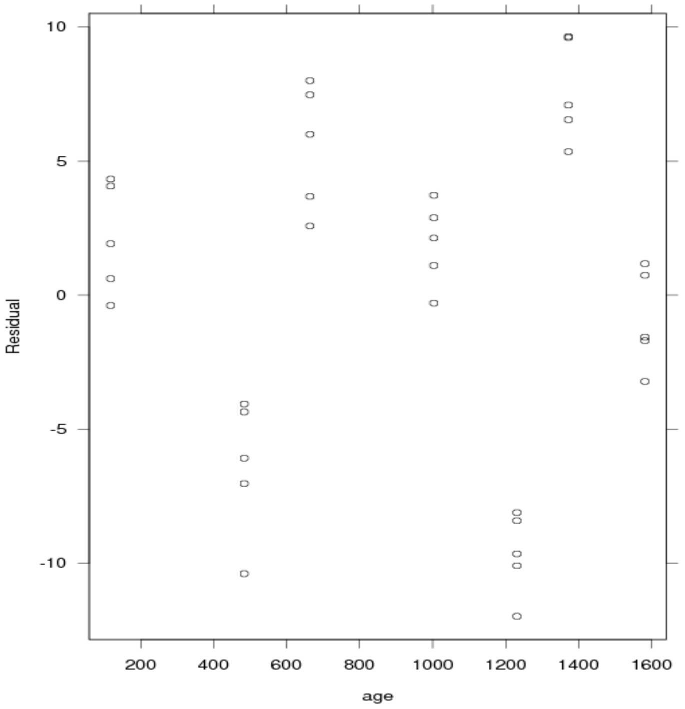

Figure 16.16: Plot of the residuals from the nonlinear model of Pinheiro and Bates

16.10 Multivariate animal genetics data - Sheep

The analysis of incomplete or unbalanced multivariate data often presents computational difficulties. These difficulties are exacerbated by either the number of random effects in the linear mixed model, the number of traits, the complexity of the variance models being fitted to the random effects or the size of the problem. In this section we illustrate two approaches to the analysis of a complex set of incomplete multivariate data.

Much of the difficulty in conducting such analyses in ASReml centres on obtaining good starting values. Derivative based algorithms such as the AI algorithm can be unreliable when fitting complex variance structures unless good starting values are available. Poor starting values may result in divergence of the algorithm or slow convergence. A particular problem with fitting unstructured variance models is keeping the estimated variance matrix positive definite. These are not simple issues and in the following we present a pragmatic approach to them.

The data are taken from a large genetic study on Coopworth lambs. A total of 5 traits, namely weaning weight (wwt), yearling weight (ywt), greasy fleece weight (gfw), fibre diameter (fdm) and ultrasound fat depth at the C site (fat) were measured on 7043 lambs. The lambs were the progeny of 92 sires and 3561 dams, produced from 4871 litters over 49 flock-year combinations. Not all traits were measured on each group. No pedigree data was available for e dams.

The aim of the analysis is to estimate heritability (h^2) of each trait and to estimate the genetic correlations between the five traits. We will present two approaches, a half-sib analysis and an analysis based on the use of an animal model, which directly defines the genetic covariance between the progeny and sires and dams.

The data fields included factors defining sire, dam and lamb (tag), covariates such as age, the age of the lamb at a set time, brr the birth rearing rank (1 = born single raised single, 2 = born twin raised single, 3 = born twin raised twin and 4 = other), sex (M, F) and grp a factor indicating the flock-year combination.

16.10.1 Half-sib analysis

In the half-sib analysis we include terms for the random effects of sires, dams and litters. In univariate analyses the variance component for sires is denoted by $\sigma_s^2 = \frac{1}{4}\sigma_A^2$ where σ_A^2 is the additive genetic variance, the variance component for dams is denoted by $\sigma_d^2 = \frac{1}{4}\sigma_A^2 + \sigma_m^2$ where σ_m^2 is the maternal variance component and the variance component for litters is denoted by σ_l^2 and represents variation attributable to the particular mating.

For a multivariate analysis these variance components for sires, dams and litters are, in theory replaced by unstructured matrices, one for each term. Additionally we assume the residuals for each trait may be correlated. Thus for this example we would like to fit a total of 4 unstructured variance models. For such a situation, it is sensible to commence

Table 16.13: REML estimates of a subset of the variance parameters for each trait for the genetic example, expressed as a ratio to their asymptotic s.e.

term	wwt	ywt	gfw	fdm	fat
sire	3.68	3.57	3.95	1.92	1.92
dam	6.25	4.93	2.78	0.37	0.05
litter	8.79	0.99	2.23	1.91	0.00
age.grp	2.29	1.39	0.31	1.15	1.74
sex.grp	2.90	3.43	3.70	-	1.83

the modelling process with a series of univariate analyses. These give starting values for the diagonals of the variance matrices, but also indicate what variance components are estimable. The ASReml job for the univariate analyses is

```
!RENAME 1 !ARG 1 2 3 4 5 #Does 5 runs one for each trait
Multivariate Sire & Dam model
!DOPART $1
!IF $1 == 1 !ASSIGN YV wwt #sets up dependent variable to each trait in turn
!IF $1 == 2 !ASSIGN YV ywt
!IF $1 == 3 !ASSIGN YV gfw
!IF $1 == 4 !ASSIGN YV fdm
!IF $1 == 5 !ASSIGN YV fat
tag
sire   92 !I
dam  3561 !I
grp    49
sex
brr     4
litter 4871
age
wwt   !M0 # !M0 identifies missing values
ywt   !M0
gfw   !M0
fdm   !M0
fat   !M0
coop.fmt
!PART 1 2 3 5
$YV ~ mu age brr sex age.sex !r idv(sire) idv(dam) idv(lit) idv(age.grp),
idv(sex.grp) !f grp #traits are substituted for $YV
!PART 4  #leaves out sex.grp for fdm
$YV ~ mu age brr sex age.sex !r idv(sire) idv(dam) idv(lit) idv(age.grp),
!f grp #$fdm is substituted for $YV
```

Tables 16.13 and 16.14 present the summary of these analyses. Fibre diameter was measured on only 2 female lambs and so interactions with sex were not fitted. The dam variance component was quite small for both fibre diameter and fat. The REML estimate of the variance component associated with litters was effectively zero for fat.

Table 16.14: Wald F statistics of the fixed effects for each trait for the genetic example

term	wwt	ywt	gfw	fdm	fat
age	331.3	67.1	52.4	2.6	7.5
brr	554.6	73.4	14.9	0.3	13.9
sex	196.1	123.3	0.2	2.9	0.6
age.sex	10.3	1.7	1.9	-	5.0

Thus in the multivariate analysis we consider fitting the following models to the sire, dam and litter effects,

$$\begin{aligned}\text{var}\left(\boldsymbol{u}_s\right) &= \boldsymbol{\Sigma}_s \otimes \boldsymbol{I}_{92}\\ \text{var}\left(\boldsymbol{u}_d\right) &= \boldsymbol{\Sigma}_d \otimes \boldsymbol{I}_{3561}\\ \text{var}\left(\boldsymbol{u}_l\right) &= \boldsymbol{\Sigma}_l \otimes \boldsymbol{I}_{4891}\end{aligned}$$

where $\boldsymbol{\Sigma}_s^{5\times5}, \boldsymbol{\Sigma}_d^{3\times3}$ and $\boldsymbol{\Sigma}_l^{4\times4}$ are positive definite symmetric matrices corresponding to the between traits variance matrices for sires, dams and litters respectively. The variance matrix for dams does not involve fibre diameter and fat depth, while the variance matrix for litters does not involve fat depth. The effects in each of the above vectors are ordered levels within traits. Lastly we assume that the residual variance matrix is given by

$$\boldsymbol{\Sigma}_e \otimes \boldsymbol{I}_{7043}$$

Table 16.15 presents the sequence variance models fitted to each of the four random terms sire, dam, litter and error in the ASReml job

```
!RENAME 1 !ARG 1 #CHANGE 1 TO 2 OR 3 FOR OTHER PATHS
Multivariate Sire & Dam
!DOPATH $1
tag
sire   92 !I
dam  3561 !I
grp    49
sex
brr     4
litter 4871
age
wwt   !MO # !MO identifies missing values
ywt   !MO
gfw   !MO
fdm   !MO
fat   !MO
!PATH 1
coop.fmt
!PATH 2
coop.fmt !CONTINUE coopmf1.rsv # uses initial values from previous .rsv file
```

```
!PATH 3
coop.fmt !CONTINUE coopmf2.rsv

!PATH 0 #SETTING UP TRAIT COMBINATIONS FOR DIFFERENT MODEL TERMS
!SUBSET TrDam123 Trait 1 2 3 0 0
!SUBSET TrLit1234 Trait 1 2 3 4 0
!SUBSET TrAG1245 Trait 1 2 4 5
!SUBSET TrSG123 Trait 1 2 3 0 0

#USING !ASSIGN TO MAKE SPECIFICATION CLEARER
#ASSIGN SIRE DAM  LITTER AND RESIDUAL INITIAL VALUES FROM UNIVARIATE ANALYSES
!ASSIGN SDIAGI !INIT 0.608 1.298 0.015 0.197 0.035 #Initial sire variances
!ASSIGN DDIAGI !INIT 2.2 4.14 0.018
!ASSIGN LDIAGI !INIT 3.74 0.97 0.019 0.941
!ASSIGN RUSI !< !INIT 9.27 0.0 16.48 0.0 0.0 0.14
0.0 0.0 0.0 3.37 0.0 0.0 0.0 0.0 1.14 !>


!ASSIGN VARF !<
 diag(TrAG1245 !INIT 0.0024 0.0019 0.0020 0.00026).age.grp,
    diag(TrSG123   !INIT 0.93 16.0 0.28).sex.grp    !>


!PART 1 #DIAGONAL FOR SIRE DAM AND LITTER UNSTRUCTURED FOR RESIDUAL
wwt ywt gfw fdm fat  ~ Trait Trait.age Trait.brr Trait.sex Trait.age.sex !r $VARF,
diag(Trait $SDIAGI).id(sire) diag(TrDam123  $DDIAGI).id(dam) diag(TrLit1234 $LDIAGI ).id(lit),
!f Trait.grp
residual id(units).us(Trait $RUSI)
!PART 2 #CHANGE DIAGONAL TO XFA1 FOR SIRE DAM AND LITTER
wwt ywt gfw fdm fat ~ Trait Trait.age Trait.brr Trait.sex Trait.age.sex !r $VARF,
xfa1(Trait).id(sire) xfa1(TrDam123).id(dam) xfa1(TrLit1234).id(lit),
!f Trait.grp mv
residual id(units).us(Trait)
!PART 3 #CHANGE XFA1 TO UNSTRUCTURED FOR SIRE AND LITTER
wwt ywt gfw fdm fat ~ Trait Trait.age Trait.brr Trait.sex Trait.age.sex !r $VARF,
us(Trait).id(sire) xfa1(TrDam123).id(dam) us(TrLit1234).id(lit),
!f Trait.grp mv
residual id(units).us(Trait)

!PART 3
VPREDICT !DEFINE
#USING !ASSIGN TO GIVE CONCISE VPREDICT
!ASSIGN lusT lit;us(TrLit1234) # us(TrLit1234).id(lit);us(TrLit1234)
!ASSIGN susT sire;us(Trait) #us(Trait).id(sire);us(Trait)
!ASSIGN uusT id(units).us(Trait);us(Trait)
X Damv  xfa1(TrDam123) # defines 54:59
F phen $uusT[1:6]+$susT[1:6]+$lusT[1:6]+xfa1(TrDam123)
# defines [1:6] elements of phen
# defines 60:65=          1:6 +             23:28 +                 44:49 + 54:59
F phen   $uusT[7:10]+$susT[7:10]+ $lusT[7:10]
# defines [7:10] elements of phen
# defines 66:69=         7:10 +              29:32 +                  50:53
F phen   $uusT[11:15]+$susT[11:15]
# defines [11:15] elements of phen
# defines 70:74=         11:15 +               33:37
F Direct  $susT *4.                #defines  75: 89= 23:37 * 4.
F Maternal Damv -$susT[1:6]       #defines  90: 95= 54:59 - 23:28
F resid phen - $susT      #defines  96:110= 60:74 - 23-37
H WWTh2 Direct[1]  phen[1]  #defines 111= 75/ 60
H YWTh2 Direct[3]  phen[3]  #defines 112= 77/ 62
H GFWh2 Direct[6]  phen[6]  #defines 113= 80/ 65
H FDMh2 Direct[10] phen[10]    #defines 114= 84/ 69
H FATh2 Direct[15] phen[15]    #defines 115= 89/ 74
R GenCor $susT        #defines 116:125 from 23:37
R MatCor Maternal              #defines 126:129 from 90:95
```

Table 16.15: Variance models fitted for each part of the ASReml job in the analysis of the genetic example

term	matrix	`!PATH 1`	`!PATH 2`	`!PATH 3`
`sire`	Σ_s	`diag`	`fa1`	`us`
`dam`	Σ_d	`diag`	`fa1`	`fa1`
`litter`	Σ_l	`diag`	`fa1`	`us`
`error`	Σ_e	`us`	`us`	`us`
LogL		-1566.45	-1488.11	-1480.89
Parameters		36	48	55

The specification in Release 3 required specification of initial values for variance parameters and also through the use of `!CONTINUE` the generation of initial values from previous analyses. In Release 4, with the functional specification and no initial values specified, ASReml will estimate initial values. In this example we start by fitting diagonal matrices for sire,dam and litter using initial values from univariate analyses and estimate an unstructured residual matrix. Unfortunately ASReml does not yet have an automatic way of taking the estimates from the univariate analyses and using them in the diagonal analysis. The Log-likelihood from this run is -20000 -1566.45. Once the model from `PATH 1` has run we can rerun the analysis changing `!ARG 1` to `!ARG 2` to obtain the next analysis. With the statement `!CONTINUE coopmf1.rsv` `ASReml` generates initial values from the `coopmf1.rsv` file, if no filename is given `ASReml` will look for the previous `.rsv` file to generate initial values. In analysis 2 we get estimates of the sire, dam and litter matrices based on a factor analysis parameterization. This can give better initial values for unstructured matrices and indicate if the estimated matrices are near singularity. The log-likelihood from this run is -20000 -1488.11. In this case the dam variance parameters are

```
 Source                Model  terms     Gamma         Sigma   Sigma/SE   % C
 xfa1(TrDam123).id(dam)       14244 effects
 TrDam123               XFA_V  0  1  0.405222      0.405222       1.30   0 P
 TrDam123               XFA_V  0  2   0.00000       0.00000       0.00   0 F
 TrDam123               XFA_V  0  3  0.616712E-02  0.616712E-02   1.14   0 P
 TrDam123               XFA_L  1  1   1.29793       1.29793       9.05   0 P
 TrDam123               XFA_L  1  2   1.68814       1.68814       9.96   0 P
 TrDam123               XFA_L  1  3  0.124492      0.124492       6.02
```

And one of the dam specific variances is zero. The resulting dam matrix is

```
Covariance/Variance/Correlation Matrix XFA xfa1(TrDam123).id(dam)
   2.090      0.8981      0.7590      0.8981
   2.190       2.845      0.8451      1.0000
  0.1613      0.2096      0.2162E-01  0.8451
   1.298       1.687      0.1243      1.0000
```

And the eigen analysis in the `.res` file is

```
Eigen Analysis of XFA matrix for xfa1(TrDam123).id(dam)
Eigen values      4.704     0.246     0.006
  Percentage     94.919     4.957     0.124
     1           0.6431   -0.7647    0.0009
     2           0.7637    0.6404   -0.0743
     3           0.0563    0.0484    0.9972
```

showing that the smallest eigenvalue is 0.006. On the basis of this ASReml with `!ARG 3`, fits unstructured matrices for sire and litter and `xfa1` for dam using initial values derived from the previous analysis in `coopmf2.rsv`. Portions of the `.asr` file from the `Path 3` run are

```
Notice: ReStartValues taken from coopmf2.rsv
Notice: LogL values are reported relative to a base of -20000.000
Notice: US matrix updates modified  1 time(s) to keep them positive definite.
Notice:   1084 singularities detected in design matrix.
  1 LogL=-1488.11     S2= 1.00000      18085 df    :  11 components restrained
  2 LogL=-1486.27     S2= 1.00000      18085 df    :   2 components restrained
  3 LogL=-1483.34     S2= 1.00000      18085 df    :   1 components restrained
  4 LogL=-1481.89     S2= 1.00000      18085 df
  5 LogL=-1481.10     S2= 1.00000      18085 df
  6 LogL=-1480.91     S2= 1.00000      18085 df
  7 LogL=-1480.89     S2= 1.00000      18085 df
  8 LogL=-1480.89     S2= 1.00000      18085 df
  9 LogL=-1480.89     S2= 1.00000      18085 df

         - - - Results from analysis of wwt ywt gfw fdm fat - - -
Notice:  US structures were modified   1 times to make them positive definite.
         If ASReml has fixed the structure [flagged by B], it may not have
            converged to a maximum likelihood solution.
         Used !EMFLAG 0 Single standard EM update when AI update unacceptable
         You could try !GU (negative definite US) or use XFA instead.

Akaike Information Criterion    43065.77 (assuming 52 parameters).
Bayesian Information Criterion  43471.52

Model_Term                             Sigma         Sigma   Sigma/SE   % C
id(units).us(Trait)           35200 effects
Trait                   US_V  1  1   9.46109       9.46109      33.29   0 P
Trait                   US_C  2  1   7.34181       7.34181      20.55   0 P
Trait                   US_V  2  2   17.6050       17.6050      27.09   0 P
Trait                   US_C  3  1  0.272536      0.272536       8.38   0 P
Trait                   US_C  3  2  0.668009      0.668009      13.99   0 P
Trait                   US_V  3  3  0.141595      0.141595      23.70   0 P
Trait                   US_C  4  1  0.963017      0.963017       2.89   0 P
Trait                   US_C  4  2   1.99771       1.99771       3.64   0 P
Trait                   US_C  4  3  0.286984      0.286984       5.08   0 P
Trait                   US_V  4  4   3.64374       3.64374       9.00   0 P
Trait                   US_C  5  1  0.850282      0.850282       8.48   0 P
Trait                   US_C  5  2   2.48313       2.48313      19.33   0 P
Trait                   US_C  5  3  0.786089E-01  0.786089E-01   7.04   0 P
Trait                   US_C  5  4  0.115894      0.115894       1.17   0 P
Trait                   US_V  5  5   1.63175       1.63175      32.90   0 P
diag(TrSG123).sex.grp          147 effects
TrSG123                DIAG_V    1   1.01106       1.01106       2.97   0 P
TrSG123                DIAG_V    2   16.0229       16.0229       3.51   0 P
TrSG123                DIAG_V    3  0.280259      0.280259       3.71   0 P
diag(TrAG1245).age.grp         196 effects
TrAG1245               DIAG_V    1  0.132755E-02  0.132755E-02   2.01   0 P
TrAG1245               DIAG_V    2  0.976533E-03  0.976533E-03   1.21   0 P
TrAG1245               DIAG_V    3  0.176684E-02  0.176684E-02   1.13   0 P
TrAG1245               DIAG_V    4  0.208076E-03  0.208076E-03   1.62   0 P
us(Trait).id(sire)                 460 effects
Trait                   US_V  1  1  0.593942      0.593942       3.68   0 P
Trait                   US_C  2  1  0.677334      0.677334       3.18   0 P
Trait                   US_V  2  2   1.55632       1.55632       3.90   0 P
Trait                   US_C  3  1  0.280482E-01  0.280482E-01   1.53   0 P
Trait                   US_C  3  2  0.287861E-02  0.287861E-02   0.10   0 P
Trait                   US_V  3  3  0.150192E-01  0.150192E-01   4.01   0 P
Trait                   US_C  4  1  0.596227E-01  0.596227E-01   0.54   0 P
Trait                   US_C  4  2 -0.657014E-01 -0.657014E-01  -0.41   0 P
Trait                   US_C  4  3  0.477561E-02  0.477561E-02   0.25   0 P
Trait                   US_V  4  4  0.157854      0.157854       1.84   0 P
Trait                   US_C  5  1  0.407282E-01  0.407282E-01   0.99   0 P
Trait                   US_C  5  2  0.133338      0.133338       1.98   0 P
Trait                   US_C  5  3  0.877122E-03  0.877122E-03   0.15   0 P
Trait                   US_C  5  4 -0.472300E-01 -0.472300E-01  -1.53   0 P
Trait                   US_V  5  5  0.326718E-01  0.326718E-01   2.00   0 P
xfa1(TrDam123).id(dam)           14244 effects
TrDam123               XFA_V  0  1  0.126746E-01  0.126746E-01   0.03   0 P
TrDam123               XFA_V  0  2   0.00000       0.00000       0.00   0 F
TrDam123               XFA_V  0  3  0.661114E-02  0.661114E-02   1.25   0 P
TrDam123               XFA_L  1  1   1.46479       1.46479       8.06   0 P
TrDam123               XFA_L  1  2   1.51911       1.51911       7.30   0 P
TrDam123               XFA_L  1  3  0.110770      0.110770       5.08   0 P
us(TrLit1234).id(lit)            19484 effects
TrLit1234               US_V  1  1   3.55275       3.55275       8.54   0 P
TrLit1234               US_C  2  1   1.53980       1.53980       3.30   0 P
TrLit1234               US_V  2  2   2.55497       2.55497       3.15   0 P
TrLit1234               US_C  3  1 -0.310141E-01 -0.310141E-01  -0.73   0 P
TrLit1234               US_C  3  2  0.450851E-01  0.450851E-01   0.74   0 P
TrLit1234               US_V  3  3  0.191030E-01  0.191030E-01   2.43   0 P
TrLit1234               US_C  4  1 -0.721026E-01 -0.721026E-01  -0.22   0 P
TrLit1234               US_C  4  2 -0.794020     -0.794020      -1.55   0 P
TrLit1234               US_C  4  3 -0.417001E-01 -0.417001E-01  -0.76   0 P
TrLit1234               US_V  4  4  0.897161      0.897161       2.29   0 P
Covariance/Variance/Correlation Matrix US Residual
  9.461      0.5689      0.2355      0.1640      0.2164
  7.342       17.60      0.4231      0.2494      0.4633
 0.2725      0.6680      0.1416      0.3995      0.1635
 0.9630       1.998      0.2870       3.644      0.4753E-01
 0.8503       2.483      0.7861E-01  0.1159       1.632
Covariance/Variance/Correlation Matrix US us(Trait).id(sire)
 0.5939      0.7045      0.2970      0.1947      0.2924
 0.6773       1.556      0.1883E-01 -0.1326      0.5913
 0.2805E-01  0.2879E-02  0.1502E-01  0.9808E-01  0.3960E-01
 0.5962E-01 -0.6570E-01  0.4776E-02  0.1579     -0.6577
 0.4073E-01  0.1333      0.8771E-03 -0.4723E-01  0.3267E-01
Covariance/Variance/Correlation Matrix XFA xfa1(TrDam123).id(dam)
  2.158      0.9961      0.8035      0.9961
  2.225       2.312      0.8066      1.0000
 0.1623      0.1687      0.1891E-01  0.8066
  1.463       1.521      0.1109      1.0000
Covariance/Variance/Correlation Matrix US us(TrLit1234).id(lit)
  3.553      0.5111     -0.1190     -0.4039E-01
  1.540       2.555      0.2041     -0.5244
-0.3101E-01  0.4509E-01  0.1910E-01 -0.3185
-0.7210E-01 -0.7940     -0.4170E-01  0.8972

                                  Wald F statistics
    Source of Variation           NumDF              F-inc
 19 Trait.age                         5              100.11
 20 Trait.brr                        15              116.72
 21 Trait.sex                         5               77.97
 23 Trait.age.sex                     4                4.17
 29 diag(TrSG123).sex.grp               147 effects fitted (      37 are zero)
 26 diag(TrAG1245).age.grp              196 effects fitted (      69 are zero)
 36 Trait.grp                           180 effects fitted (+      65 singular)
 31 us(Trait).sire                      460 effects fitted (      20 are zero)
 33 xfa1(TrDam123).dam                10683 effects fitted (       8 are zero)
 35 us(TrLit1234).lit                 19484 effects fitted (      20 are zero)
```

The REML estimates of all the variance matrices except for the dam components are positive definite. Heritabilities for each trait can be calculated using the `VPREDICT` facility of ASReml. The heritability is given by

$$h^2 = \frac{\sigma^2_A}{\sigma^2_P}$$

where σ^2_P is the phenotypic variance and is given by

$$\sigma^2_P = \sigma^2_s + \sigma^2_d + \sigma^2_l + \sigma^2_e$$

recalling that

$$\begin{aligned} \sigma^2_s &= \frac{1}{4}\sigma^2_A \\ \sigma^2_d &= \frac{1}{4}\sigma^2_A + \sigma^2_m \end{aligned}$$

In the half-sib analysis we only use the estimate of additive genetic variance from the sire variance component. ASReml then carries out the `VPREDICT` instructions in the `.asr` file, stores the instructions in a `.pin` file and produces the following output in a `.pvc` file.

```
ASReml 5.0 [01 Apr 2014] Multivariate Sire & Dam
          coopmf3.pvc created 16 Nov 2014 21:51:35.702

          - - - Results from analysis of wwt ywt gfw fdm fat - - -

 id(units).us(Trait)          35200 effects
   1 id(units).us(Trait);us(Trait)           V  1  1      9.46109        0.284202
   2 id(units).us(Trait);us(Trait)           C  2  1      7.34181        0.357266
   3 id(units).us(Trait);us(Trait)           V  2  2      17.6050        0.649871
   4 id(units).us(Trait);us(Trait)           C  3  1     0.272536        0.325222E-01
   5 id(units).us(Trait);us(Trait)           C  3  2     0.668009        0.477490E-01
   6 id(units).us(Trait);us(Trait)           V  3  3     0.141595        0.597447E-02
   7 id(units).us(Trait);us(Trait)           C  4  1     0.963017        0.333224
   8 id(units).us(Trait);us(Trait)           C  4  2      1.99771        0.548821
   9 id(units).us(Trait);us(Trait)           C  4  3     0.286984        0.564929E-01
  10 id(units).us(Trait);us(Trait)           V  4  4      3.64374        0.404860
  11 id(units).us(Trait);us(Trait)           C  5  1     0.850282        0.100269
  12 id(units).us(Trait);us(Trait)           C  5  2      2.48313        0.128460
  13 id(units).us(Trait);us(Trait)           C  5  3     0.786089E-01    0.111660E-01
  14 id(units).us(Trait);us(Trait)           C  5  4     0.115894        0.990547E-01
  15 id(units).us(Trait);us(Trait)           V  5  5      1.63175        0.495973E-01
 diag(TrSG123).sex.grp          147 effects
  16 diag(TrSG123).sex.grp;diag(TrSG123)     V    1       1.01106        0.340424
  17 diag(TrSG123).sex.grp;diag(TrSG123)     V    2       16.0229         4.56493
  18 diag(TrSG123).sex.grp;diag(TrSG123)     V    3      0.280259        0.755415E-01
 diag(TrAG1245).age.grp         196 effects
  19 diag(TrAG1245).age.grp;diag(TrAG1245)   V    1      0.132755E-02    0.660473E-03
  20 diag(TrAG1245).age.grp;diag(TrAG1245)   V    2      0.976533E-03    0.807052E-03
  21 diag(TrAG1245).age.grp;diag(TrAG1245)   V    3      0.176684E-02    0.156358E-02
  22 diag(TrAG1245).age.grp;diag(TrAG1245)   V    4      0.208076E-03    0.128442E-03
 us(Trait).id(sire)             460 effects
  23 us(Trait).id(sire);us(Trait)            V  1  1     0.593942        0.161397
  24 us(Trait).id(sire);us(Trait)            C  2  1     0.677334        0.212998
  25 us(Trait).id(sire);us(Trait)            V  2  2      1.55632        0.399056
  26 us(Trait).id(sire);us(Trait)            C  3  1     0.280482E-01    0.183322E-01
  27 us(Trait).id(sire);us(Trait)            C  3  2     0.287861E-02    0.287861E-01
  28 us(Trait).id(sire);us(Trait)            V  3  3     0.150192E-01    0.374544E-02
  29 us(Trait).id(sire);us(Trait)            C  4  1     0.596227E-01    0.110412
  30 us(Trait).id(sire);us(Trait)            C  4  2    -0.657014E-01   -0.410000
  31 us(Trait).id(sire);us(Trait)            C  4  3     0.477561E-02    0.191024E-01
  32 us(Trait).id(sire);us(Trait)            V  4  4     0.157854        0.857902E-01
  33 us(Trait).id(sire);us(Trait)            C  5  1     0.407282E-01    0.411396E-01
  34 us(Trait).id(sire);us(Trait)            C  5  2     0.133338        0.673424E-01
  35 us(Trait).id(sire);us(Trait)            C  5  3     0.877122E-03    0.584748E-02
  36 us(Trait).id(sire);us(Trait)            C  5  4    -0.472300E-01    -1.53000
  37 us(Trait).id(sire);us(Trait)            V  5  5     0.326718E-01    0.163359E-01
 xfa1(TrDam123).id(dam)       14244 effects
  38 xfa1(TrDam123).id(dam);xfa1(TrDam123)   V  0  1     0.126746E-01    0.422487
  39 xfa1(TrDam123).id(dam);xfa1(TrDam123)   V  0  2      0.00000         0.00000
  40 xfa1(TrDam123).id(dam);xfa1(TrDam123)   V  0  3     0.661114E-02    0.528891E-02
  41 xfa1(TrDam123).id(dam);xfa1(TrDam123)   L  1  1      1.46479        0.181736
  42 xfa1(TrDam123).id(dam);xfa1(TrDam123)   L  1  2      1.51911        0.208097
  43 xfa1(TrDam123).id(dam);xfa1(TrDam123)   L  1  3     0.110770        0.218051E-01
 us(TrLit1234).id(lit)        19484 effects
  44 us(TrLit1234).id(lit);us(TrLit1234)     V  1  1      3.55275        0.416013
```

```
45 us(TrLit1234).id(lit);us(TrLit1234)     C  2  1      1.53980        0.466606
46 us(TrLit1234).id(lit);us(TrLit1234)     V  2  2      2.55497        0.811102
47 us(TrLit1234).id(lit);us(TrLit1234)     C  3  1    -0.310141E-01   -0.730000
48 us(TrLit1234).id(lit);us(TrLit1234)     C  3  2     0.450851E-01    0.609258E-01
49 us(TrLit1234).id(lit);us(TrLit1234)     V  3  3     0.191030E-01    0.786132E-02
50 us(TrLit1234).id(lit);us(TrLit1234)     C  4  1    -0.721026E-01   -0.220000
51 us(TrLit1234).id(lit);us(TrLit1234)     C  4  2    -0.794020        -1.55000
52 us(TrLit1234).id(lit);us(TrLit1234)     C  4  3    -0.417001E-01   -0.760000
53 us(TrLit1234).id(lit);us(TrLit1234)     V  4  4     0.897161        0.391773
54 Damv                      2.1583       0.33589
55 Damv                      2.2252       0.37368
56 Damv                      2.3077       0.63232
57 Damv                     0.16225       0.32785E-01
58 Damv                     0.16827       0.47001E-01
59 Damv                     0.18881E-01   0.59274E-02
60 phen  1                   13.620       0.57871
61 phen  2                   9.5589       0.48024
62 phen  3                   21.723       0.84528
63 phen  4                   1.7344       0.17740
64 phen  5                   2.2351       0.18402
65 phen  6                  0.28649       0.20501E-01
66 phen  7                  0.95054       0.29825
67 phen  8                   1.1380       0.37755
68 phen  9                  0.25006       0.37255E-01
69 phen 10                   4.6988       0.22522
70 phen 11                  0.89101       0.10759
71 phen 12                   2.6165       0.14261
72 phen 13                  0.79486E-01   0.12431E-01
73 phen 14                  0.68664E-01   0.10198
74 phen 15                   1.6644       0.51205E-01
75 Direct 23                 2.3758       0.64586
76 Direct 24                 2.7093       0.85213
77 Direct 25                 6.2253        1.5966
78 Direct 26                0.11219       0.73359E-01
79 Direct 27                0.11514E-01   0.11109
80 Direct 28                0.60077E-01   0.14996E-01
81 Direct 29                0.23849       0.44400
82 Direct 30               -0.26281       0.64674
83 Direct 31                0.19102E-01   0.76604E-01
84 Direct 32                0.63142       0.34354
85 Direct 33                0.16291       0.16518
86 Direct 34                0.53335       0.27002
87 Direct 35                0.35085E-02   0.23889E-01
88 Direct 36               -0.18892       0.12314
89 Direct 37                0.13069       0.65488E-01
90 Maternal 54               1.5643       0.37542
91 Maternal 55               1.5478       0.43280
92 Maternal 56              0.75138       0.75145
93 Maternal 57              0.13421       0.37770E-01
94 Maternal 58              0.16539       0.54770E-01
95 Maternal 59              0.38619E-02   0.70075E-02
96 resid 60                  13.027       0.55777
97 resid 61                  8.8816       0.43344
98 resid 62                  20.167       0.75071
99 resid 63                  1.7063       0.17660
```

```
100 resid 64                  2.2322       0.18225
101 resid 65                 0.27147       0.20194E-01
102 resid 66                 0.89091       0.28008
103 resid 67                  1.2037       0.34725
104 resid 68                 0.24528       0.32775E-01
105 resid 69                  4.5409       0.21411
106 resid 70                 0.85028       0.10023
107 resid 71                  2.4831       0.12849
108 resid 72                 0.78609E-01   0.11170E-01
109 resid 73                 0.11589       0.99338E-01
110 resid 74                  1.6318       0.49595E-01
    WWTh2        = Direct 2  75/phen  1   60=          0.1744    0.0460
    YWTh2        = Direct 2  77/phen  3   62=          0.2866    0.0691
    GFWh2        = Direct 2  80/phen  6   65=          0.2097    0.0519
    FDMh2        = Direct 3  84/phen 10   69=          0.1344    0.0713
    FATh2        = Direct 3  89/phen 15   74=          0.0785    0.0388
    GenCor  2  1 = us(Tr 24/SQR[us(Tr 23*us(Tr 25]=    0.7045    0.1024
    GenCor  3  1 = us(Tr 26/SQR[us(Tr 23*us(Tr 28]=    0.2970    0.1720
    GenCor  3  2 = us(Tr 27/SQR[us(Tr 25*us(Tr 28]=    0.0188    0.1808
    GenCor  4  1 = us(Tr 29/SQR[us(Tr 23*us(Tr 32]=    0.1947    0.3521
    GenCor  4  2 = us(Tr 30/SQR[us(Tr 25*us(Tr 32]=   -0.1326    0.3249
    GenCor  4  3 = us(Tr 31/SQR[us(Tr 28*us(Tr 32]=    0.0981    0.3874
    GenCor  5  1 = us(Tr 33/SQR[us(Tr 23*us(Tr 37]=    0.2924    0.2747
    GenCor  5  2 = us(Tr 34/SQR[us(Tr 25*us(Tr 37]=    0.5913    0.2026
    GenCor  5  3 = us(Tr 35/SQR[us(Tr 28*us(Tr 37]=    0.0396    0.2687
    GenCor  5  4 = us(Tr 36/SQR[us(Tr 32*us(Tr 37]=   -0.6577    0.3854
    MatCor  2  1 = Mater 91/SQR[Mater 90*Mater 92]=    1.4277    0.5305
    MatCor  3  1 = Mater 93/SQR[Mater 90*Mater 95]=    1.7267    1.4388
    MatCor  3  2 = Mater 94/SQR[Mater 92*Mater 95]=    3.0703    2.9688
Notice: The parameter estimates are followed by
         their approximate standard errors.
```

16.10.2 Animal model

In this section we will illustrate the use of a pedigree file to define the genetic relationships between animals. This is an alternate method of estimating additive genetic variance for these data. The data file has been modified by adding 10000 to the dam ID (now 10001:13561) so that the lamb, sire and dam ID's are distinct. They appear as the first genetic relationships are available for this data so the data file doubles as the pedigree file. The multi-trait additive genetic variance matrix, $\mathbf{\Sigma}_A$ of the animals (sires, dams and lambs) is given by

$$\text{var}\,(\boldsymbol{u}_A) = \mathbf{\Sigma}_A \otimes \boldsymbol{A}$$

where $\boldsymbol{A}$ is the genetic relationship matrix and $\boldsymbol{u}_A$ are the trait BLUPs ordered animals within traits There are a total of $10696 = 92 + 3561 + 7043$ animals in the pedigree.

Multivariate analysis involving several strata (here `animal` (direct/additive genetic), `dam` (maternal) and `litter`) typically involves several runs. The ASReml input file presented below has five parts which show the use of `FA` structures to get initial values for estimation of unstructured matrices, and their use when estimated unstructured matrices are not positive definite as is the case with the tag matrix here, but omits earlier runs involved with linear model selection and obtaining initial values. This model is not equivalent to the sire/dam/litter model with respect to the animal/litter components for gfw, fd and fat.

```
!RENAME 1 !ARG 1 #CHANGE 1 TO 2,3,4 OR 5 FOR OTHER PATHS
Multivariate Animal model
!DOPART $1
tag !P
sire   92 !I
dam  !P
grp    49
sex
brr     4
litter 4871
age
wwt   !MO # !MO identifies missing values
ywt   !MO
gfw   !MO
fdm   !MO
fat   !MO
pcoop.fmt # read pedigree from first three fields
!PATH 1  // pcoop.fmt
!PATH 2  // pcoop.fmt !CONTINUE pcoopf1.rsv !MAXI  40
!PATH 3  // pcoop.fmt !CONTINUE pcoopf2.rsv !MAXI  40
!PATH 4  // pcoop.fmt !CONTINUE pcoopf2.rsv !MAXI  40
!PATH 5  // pcoop.fmt !CONTINUE pcoopf4.rsv !MAXI  40


!PART 0
!SUBSET TrDam12 Trait 1 2 0 0 0
!SUBSET TrLit1234 Trait 1 2 3 4 0
!SUBSET TrAG1245 Trait 1 2 4 5
!SUBSET TrSG123 Trait 1 2 3 0 0
!SUBSET TrDa123  Trait 1 2 3 0 0


#USING !ASSIGN TO MAKE SPECIFICATION CLEARER
!ASSIGN TDIAGI  !INIT 2.3759    6.2256        0.60075E-01   0.63086    0.13069  !GP
!ASSIGN DDIAGI  !INIT 2.1584    2.3048 !GP
!ASSIGN LDIAGI  !INIT 3.55265   2.55777 0.191238E-01   0.897272   !GP
!ASSIGN RUSI    !< !INIT 13.390        9.0747      17.798 0.31961 0.87272 0.13452
0.71374    1.4028  0.23141  4.0677  0.72812  2.0831  0.75977E-01  0.25782 1.5337 !GP !>

!ASSIGN VARF !<
 diag(TrAG1245 !INIT 0.0024 0.0019 0.0020 0.00026).age.grp,
    diag(TrSG123   !INIT 0.93 16.0 0.28).sex.grp    !>



!PATH 1
wwt ywt gfw fdm fat ~ Trait Trait.age Trait.brr Trait.sex Trait.age.sex !r $VARF,
diag(Trait $TDIAGI).nrm(tag) diag(TrDam12 $DDIAGI).nrm(dam) diag(TrLit1234 $LDIAGI).id(lit),
!f Trait.grp
residual id(units).us(Trait $RUSI)
```

```
!PATH 2
wwt ywt gfw fdm fat  ~ Trait Trait.age Trait.brr Trait.sex Trait.age.sex !r $VARF,
xfa1(Trait).nrm(tag) xfa1(TrDam12).nrm(dam) xfa1(TrLit1234).id(lit),
!f Trait.grp
residual id(units).us(Trait)

!PATH 3
wwt ywt gfw fdm fat ~ Trait Trait.age Trait.brr Trait.sex Trait.age.sex !r $VARF,
us(Trait).nrm(tag) xfa1(TrDam12).nrm(dam) us(TrLit1234).id(lit),
!f Trait.grp
residual id(units).us(Trait)

!PATH 4
wwt ywt gfw fdm fat ~ Trait Trait.age Trait.brr Trait.sex Trait.age.sex !r $VARF,
xfa2(Trait).nrm(tag) xfa1(TrDam12).nrm(dam) us(TrLit1234).id(lit),
!f Trait.grp
residual id(units).us(Trait)

!PART 5
wwt ywt gfw fdm fat ~ Trait Trait.age Trait.brr Trait.sex Trait.age.sex !r $VARF,
xfa3(Trait).nrm(tag) xfa1(TrDam12).nrm(dam) us(TrLit1234).id(lit),
!f Trait.grp
residual id(units).us(Trait)
```

The term `function(Tr).nrm(tag)` now replaces the `function(Tr).id(sire)` and picks up part of `function(TrDam12).id(dam)` variation present in the half-sib analysis. This analysis uses information from both sires and dams to estimate additive genetic variance. The dam variance component is this analysis estimates the maternal variance component. It is only significant for the weaning and yearling weights. The litter variation remains unchanged. Notice again how the maternal effect is only fitted for the first 2 trait and the litter effect for the first 4 traits. The critical detail is that SUBSET is used to setup `TrDam12`, a variable using the first two traits. ASReml uses the relationship matrix for the `dam` dimension [1] since `dam` is defined with `!P`. In this case there is no difference between fitting `nrm(dam)` and `id(ide(dam))` since there is no pedigree information on dams. It is preferable to be explicit (specify `nrm(dam)` when the relationship matrix is required, and `id(ide(dam))` in the G structure definition).

In this case `PATH` 1, 2 and 3 were run in turn but in `PATH 3` ASReml had trouble converging because in each iteration the unstructured `us(tag)` matrix is not positive definite and so ASReml uses a slower EM algorithm that keeps the estimates in the parameter space but the convergence is very slow. Here is the convergence log for `PATH 3`

```
 Notice:  15358 singularities detected in design matrix.
   1 LogL=-1543.55      S2= 1.00000      18085 df    :  15 components restrained
 Notice: US matrix updates modified  1 time(s) to keep them positive definite.
   2 LogL=-1540.93      S2= 1.00000      18085 df    :  15 components restrained
 Notice: US matrix updates modified  1 time(s) to keep them positive definite.
: :       :            :
  38 LogL=-1538.34      S2= 1.00000      18085 df    :  15 components restrained
```

[1] reported in the .asr file

```
  39 LogL=-1538.33     S2= 1.00000      18085 df    :  14 components restrained
  40 LogL=-1538.32     S2= 1.00000      18085 df    :  15 components restrained
```

To avoid this problem in `PATH 4` and `5` we use `xfa2` and `xfa3` structures. These converge much faster. Here is the convergence log and resulting estimates for `PATH 5`

```
Notice: ReStartValues taken from pcoopf4.rsv
 Notice: LogL values are reported relative to a base of -20000.000

    Note:  XFA model: lower loadings initially held fixed.
 Notice:  29764 singularities detected in design matrix.
   1 LogL=-1558.44     S2= 1.00000      18085 df    :   1 components restrained
   2 LogL=-1541.77     S2= 1.00000      18085 df    :   8 components restrained
   3 LogL=-1538.27     S2= 1.00000      18085 df    :   1 components restrained
   4 LogL=-1534.53     S2= 1.00000      18085 df    :   1 components restrained
   5 LogL=-1532.53     S2= 1.00000      18085 df    :   1 components restrained
   6 LogL=-1531.90     S2= 1.00000      18085 df    :   1 components restrained

    Note: XFA model fitted with rotation.
   7 LogL=-1531.73     S2= 1.00000      18085 df    :   1 components restrained
   8 LogL=-1531.66     S2= 1.00000      18085 df
   9 LogL=-1531.64     S2= 1.00000      18085 df
  10 LogL=-1531.64     S2= 1.00000      18085 df

          - - - Results from analysis of wwt ywt gfw fdm fat - - -
 Akaike Information Criterion    43151.28 (assuming 44 parameters).
 Bayesian Information Criterion  43494.60

 Model_Term                             Sigma         Sigma   Sigma/SE   % C
 id(units).us(Trait)          35200 effects
 Trait                   US_V  1  1   8.73848       8.73848      30.29   0 P
 Trait                   US_C  2  1   7.28418       7.28418      20.19   0 P
 Trait                   US_V  2  2   17.7519       17.7519      26.87   0 P
 Trait                   US_C  3  1  0.247701      0.247701       5.87   0 P
 Trait                   US_C  3  2  0.705206      0.705206      14.31   0 P
 Trait                   US_V  3  3  0.109534      0.109534      11.21   0 P
 Trait                   US_C  4  1  0.816946      0.816946       2.22   0 P
 Trait                   US_C  4  2   2.03823       2.03823       3.68   0 P
 Trait                   US_C  4  3  0.252623      0.252623       3.82   0 P
 Trait                   US_V  4  4   3.31364       3.31364       7.50   0 P
 Trait                   US_C  5  1  0.871291      0.871291       6.95   0 P
 Trait                   US_C  5  2   2.53142       2.53142      19.25   0 P
 Trait                   US_C  5  3  0.821032E-01  0.821032E-01   4.52   0 P
 Trait                   US_C  5  4  0.208739      0.208739       1.60   0 P
 Trait                   US_V  5  5   1.54280       1.54280      24.00   0 P
 diag(TrSG123).sex.grp          147 effects
 TrSG123                DIAG_V    1   1.01250       1.01250       2.96   0 P
 TrSG123                DIAG_V    2   15.2159       15.2159       3.49   0 P
 TrSG123                DIAG_V    3  0.279183      0.279183       3.71   0 P
 diag(TrAG1245).age.grp         196 effects
 TrAG1245               DIAG_V    1  0.142096E-02  0.142096E-02   2.04   0 P
 TrAG1245               DIAG_V    2  0.143897E-02  0.143897E-02   1.54   0 P
 TrAG1245               DIAG_V    3  0.163778E-02  0.163778E-02   1.10   0 P
 TrAG1245               DIAG_V    4  0.207274E-03  0.207274E-03   1.61   0 P
```

```
us(TrLit1234).id(lit)        19484 effects
TrLit1234               US_V  1  1   3.84738       3.84738          9.19   0 P
TrLit1234               US_C  2  1   2.52256       2.52256          5.47   0 P
TrLit1234               US_V  2  2   4.07860       4.07860          5.46   0 P
TrLit1234               US_C  3  1  0.767402E-01  0.767402E-01   2.05   0 P
TrLit1234               US_C  3  2  0.206265      0.206265         4.36   0 P
TrLit1234               US_V  3  3  0.250400E-01  0.250400E-01   3.30   0 P
TrLit1234               US_C  4  1 -0.118244     -0.118244        -0.35   0 P
TrLit1234               US_C  4  2 -0.824135     -0.824135        -1.58   0 P
TrLit1234               US_C  4  3 -0.492320E-01 -0.492320E-01  -0.85   0 P
TrLit1234               US_V  4  4  0.704947      0.704947         1.74   0 P
xfa1(TrDam12).id(dam)        32088 effects
TrDam12                XFA_V  0  1   0.00000       0.00000          0.00   0 F
TrDam12                XFA_V  0  2   0.00000       0.00000          0.00   0 F
TrDam12                XFA_L  1  1   1.27045       1.27045         10.00   0 P
TrDam12                XFA_L  1  2   1.15350       1.15350          5.66   0 P
xfa3(Trait).nrm(tag)         85568 effects
Trait                  XFA_V  0  1   0.00000       0.00000          0.00   0 F
Trait                  XFA_V  0  2   0.00000       0.00000          0.00   0 F
Trait                  XFA_V  0  3   0.00000       0.00000          0.00   0 F
Trait                  XFA_V  0  4  0.423585      0.423585         1.21   0 P
Trait                  XFA_V  0  5   0.00000       0.00000          0.00   0 B
Trait                  XFA_L  1  1 -0.109659E-02 -0.109659E-02   0.00   0 F
Trait                  XFA_L  1  2 -0.180117     -0.180117        -2.88   0 P
Trait                  XFA_L  1  3  0.219215      0.219215         3.53   0 P
Trait                  XFA_L  1  4  0.214461E-01  0.214461E-01   0.07   0 P
Trait                  XFA_L  1  5  0.177932      0.177932         1.18   0 P
Trait                  XFA_L  2  1   1.17261       1.17261          0.00   0 F
Trait                  XFA_L  2  2  0.530954E-01  0.530954E-01   0.00   0 F
Trait                  XFA_L  2  3  0.604977E-01  0.604977E-01   1.31   0 P
Trait                  XFA_L  2  4  0.286377      0.286377         0.99   0 P
Trait                  XFA_L  2  5 -0.460967E-01 -0.460967E-01  -0.33   0 P
Trait                  XFA_L  3  1 -0.123499     -0.123499        -0.28   0 P
Trait                  XFA_L  3  2 -0.938092E-01 -0.938092E-01  -1.09   0 P
Trait                  XFA_L  3  3  0.115989      0.115989         1.12   0 P
Trait                  XFA_L  3  4  0.439945      0.439945         1.40   0 P
Trait                  XFA_L  3  5 -0.288612     -0.288612        -2.62   0 P
tag                    NRM   10696
Warning: Code B - fixed at a boundary (!GP)       F - fixed by user
              ? - liable to change from P to B    P - positive definite
              C - Constrained by user (!VCC)      U - unbounded
              S - Singular Information matrix
S means there is no information in the data for this parameter.
Very small components with Comp/SE ratios of zero sometimes indicate poor
          scaling.  Consider rescaling the design matrix in such cases.
Covariance/Variance/Correlation Matrix US Residual
  8.738      0.5848      0.2532      0.1518      0.2373
  7.284       17.75      0.5057      0.2658      0.4837
 0.2477      0.7052      0.1095      0.4193      0.1997
 0.8169       2.038      0.2526       3.314      0.9232E-01
 0.8713       2.531      0.8210E-01  0.2087       1.543
Covariance/Variance/Correlation Matrix US us(TrLit1234).id(lit
  3.847      0.6368      0.2472     -0.7180E-01
  2.523       4.079      0.6454     -0.4860
 0.7674E-01  0.2063      0.2504E-01 -0.3706
```

```
-0.1182     -0.8241     -0.4923E-01  0.7049
Covariance/Variance/Correlation Matrix XFA xfa1(TrDam12).id(dam
  1.614      1.0000      1.0000
  1.465       1.330      1.0000
  1.270       1.153      1.0000
Covariance/Variance/Correlation Matrix XFA xfa3(Trait).nrm(tag)
  1.389      0.2978      0.1871      0.2861     -0.4630E-01 -0.9303E-03  0.9948     -0.1017
 0.7379E-01  0.4419E-01 -0.8809     -0.1709     -0.1009     -0.8568      0.2526     -0.4495
 0.5629E-01 -0.4726E-01  0.6514E-01  0.3410      0.3155E-01  0.8583      0.2355      0.4560
 0.2820     -0.3004E-01  0.7277E-01  0.6992     -0.4761      0.2416E-01  0.3414      0.5260
-0.1869E-01 -0.7261E-02  0.2757E-02 -0.1363      0.1173      0.5210     -0.1323     -0.8432
-0.1097E-02 -0.1801      0.2190      0.2020E-01  0.1784      1.0000       0.000       0.000
  1.173      0.5310E-01  0.6011E-01  0.2855     -0.4530E-01   0.000      1.0000       0.000
-0.1199     -0.9449E-01  0.1164      0.4398     -0.2888       0.000       0.000      1.0000
```

Note that the XFA matrix associated with tag has 8 rows (and columns) the first five relate to the five traits and the last three relate to the three factors.

Bibliography

Breslow, N. E. (2003). Whither PQL?, *Technical Report 192*, UW Biostatistics Working Paper Series, University of Washington.
URL: *http://www.bepress.com/uwbiostat/paper192/*

Breslow, N. E. and Clayton, D. G. (1993). Approximate inference in generalized linear mixed models, *Journal of the American Statistical Association* **88**: 9–25.

Breslow, N. E. and Lin, X. (1995). Bias correction in generalised linear mixed models with a single component of dispersion, *Biometrika* **82**: 81–91.

Cox, D. R. and Hinkley, D. V. (1974). *Theoretical Statistics*, Chapman and Hall.

Cox, D. R. and Snell, E. J. (1981). *Applied Statistics; Principals and Examples*, Chapman and Hall.

Cressie, N. A. C. (1991). *Statistics for spatial data*, John Wiley and Sons.

Cullis, B. R. and Gleeson, A. C. (1991). Spatial analysis of field experiments - an extension to two dimensions, *Biometrics* **47**: 1449–1460.

Cullis, B. R., Gleeson, A. C., Lill, W. J., Fisher, J. A. and Read, B. J. (1989). A new procedure for the analysis of early generation variety trials, *Applied Statistics* **38**: 361–375.

Cullis, B. R., Gogel, B. J., Verbyla, A. P. and Thompson, R. (1998). Spatial analysis of multi-environment early generation trials, *Biometrics* **54**: 1–18.

Dempster, A. P., Selwyn, M. R., Patel, C. M. and Roth, A. J. (1984). Statistical and computational aspects of mixed model analysis., *Applied Statistics* **33**: 203–214.

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

Fernando, R. and Grossman, M. (1990). Genetic evaluation with autosomal and x-chromosomal inheritance, *Theoretical and Applied Genetics* **80**: 75–80.

Gilmour, A. R. (2007). Mixed model regression mapping for qtl detection in experimental crosses., *Computational Statistics and Data Analysis* **51**: 3749–3764.

Gilmour, A. R., Cullis, B. R. and Verbyla, A. P. (1997). Accounting for natural and extraneous variation in the analysis of field experiments, *Journal of Agricultural, Biological, and Environmental Statistics* **2**: 269–273.

Gilmour, A. R., Cullis, B. R., Welham, S. J., Gogel, B. J. and Thompson, R. (2004). An efficient computing strategy for prediction in mixed linear models, *Computational Statistics and Data Analysis* **44**: 571–586.

Gilmour, A. R., Thompson, R. and Cullis, B. R. (1995). AI, an efficient algorithm for REML estimation in linear mixed models, *Biometrics* **51**: 1440–1450.

Gleeson, A. C. and Cullis, B. R. (1987). Residual maximum likelihood (REML) estimation of a neighbour model for field experiments, *Biometrics* **43**: 277–288.

Gogel, B. J. (1997). *Spatial analysis of multi-environment variety trials*, PhD thesis, Department of Statistics, University of Adelaide.

Goldstein, H. and Rasbash, J. (1996). Improved approximations for multilevel models with binary response, *Journal of the Royal Statistical Society A – General* **159**: 505–513.

Goldstein, H., Rasbash, J., Plewis, I., Draper, D., Browne, W., Yang, M., Woodhouse, G. and Healy, M. (1998). *A user's guide to MLwiN*, Institute of Education, London.
URL: *http://multilevel.ioe.ac.uk/*

Green, P. J. and Silverman, B. W. (1994). *Nonparametric regression and generalized linear models*, Chapman and Hall.

Harvey, W. R. (1977). *Users' guide to LSML76*, The Ohio State University, Columbus.

Harville, D. A. (1997). *Matrix Algebra from a statisticians perspective.*, Springer-Verlaag.

Harville, D. and Mee, R. (1984). A mixed model procedure for analysing ordered categorical data, *Biometrics* **40**: 393–408.

Haskard, K. A. (2006). *Anisotropic Matérn correlation and other issues in model-based geostatistics*, PhD thesis, BiometricsSA, University of Adelaide.

Kammann, E. E. and Wand, M. P. (2003). Geoadditive models, *Applied Statistics* **52**(1): 1–18.

Keen, A. (1994). Procedure IRREML, *GLW-DLO Procedure Library Manual*, Agricultural Mathematics Group, Wageningen, The Netherlands, pp. Report LWA–94–16.

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

Lane, P. W. and Nelder, J. A. (1982). Analysis of covariance and standardisation as instances of predicton, *Biometrics* **38**: 613–621.

McCulloch, C. and Searle, S. R. (2001). *Generalized, Linear, and Mixed Models*, Wiley.

Meuwissen and Lou (1992). Forming iniverse nrm, *Genetics, Selection and Evolution* **24**: 305–313.

Millar, R. and Willis, T. (1999). Estimating the relative density of snapper in and around a marine reserve using a log-linear mixed-effects model, *Australian and New Zealand Journal of Statistics* **41**: 383–394.

Nelder, J. A. (1994). The statistics of linear models: back to basics, *Statistics and Computing* **4**: 221–234.

Patterson, H. D. and Thompson, R. (1971). Recovery of interblock information when block sizes are unequal, *Biometrika* **31**: 100–109.

Pinheiro, J. C. and Bates, D. M. (2000). *Mixed-Effects Models in S and S-PLUS.*, Springer-Verlaag.

Quaas, R. L. (1976). Computing the diagonal elements and inverse of a large numerator relationship matrix., *Biometrics* **32**: 949–953.

Robinson, G. K. (1991). That blup is a good thing: The estimation of random effects, *Statistical Science* **6**: 15–51.

Rodriguez, G. and Goldman, N. (2001). Improved estimation procedures for multilevel models with binary response: A case study, *Journal of the Royal Statistical Society A – General* **164**(2): 339–355.

Sargolzaei, Iwaisaki and Colleau (2005). A fast algorithm for computing inbreeding coefficients in large populations, *Genetics, Selection and Evolution* **122**: 325–331.

Schall, R. (1991). Estimation in generalized linear models with random effects, *Biometrika* **78**(4): 719–27.

Searle, S. R. (1971). *Linear Models*, New York: John Wiley and Sons, Inc.

Searle, S. R. (1982). *Matrix algebra useful for statistics*, New York: John Wiley and Sons, Inc.

Searle, S. R., Casella, G. and McCulloch, C. E. (1992). *Variance Components*, New York: John Wiley and Sons, Inc.

Self, S. C. and Y., L. K. (1987). Asymptotic properties of maximum likelihood estimators and likelihood ratio tests under non-standard conditions., *Journal of the American Statistical Society* **82**: 605–610.

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

Smith, A., Cullis, B. R. and Thompson, R. (2001). Analysing variety by environment data using multiplicative mixed models and adjustments for spatial field trend, *Biometrics* **57**: 1138–1147.

Smith, A., Cullis, B. R. and Thompson, R. (2005). The analysis of crop cultivar breeding and evaluation trials: an overview of current mixed model approaches [review], *Journal of Agricultural Science* **143**: 449–462.

Stein, M. L. (1999). *Interpolation of Spatial Data: Some Theory for Kriging*, Springer-Verlag, New York.

Stevens, M. M., Fox, K. M., Warren, G. N., Cullis, B. R., Coombes, N. E. and Lewin, L. G. (1999). An image analysis technique for assessing resistance in rice cultivars to root-feeding chironomid midge larvae (diptera: Chironomidae), *Field Crops Research* **66**: 25–26.

Stroup, W. W., Baenziger, P. S. and Mulitze, D. K. (1994). Removing spatial variation from wheat yield trials: a comparison of methods, *Crop. Sci* **86**: 62–66.

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

Thompson, R., Cullis, B., Smith, A. and Gilmour, A. (2003). A sparse implementation of the average information algorithm for factor analytic and reduced rank variance models, *Australian and New Zealand Journal of Statistics* **45**: 445–459.

Verbyla, A. P. (1990). A conditional derivation of residual maximum likelihood, *Australian Journal of Statistics* **32**: 227–230.

Verbyla, A. P., Cullis, B. R., Kenward, M. G. and Welham, S. J. (1999). The analysis of designed experiments and longitudinal data by using smoothing splines (with discussion), *Applied Statistics* **48**: 269–311.

Waddington, D., Welham, S. J., Gilmour, A. R. and Thompson, R. (1994). Comparisons of some glmm estimators for a simple binomial model., *Genstat Newsletter* **30**: 13–24.

Welham, S. J. (2005). Glmm fits a generalized linear mixed model., *in* R. Payne and P. Lane (eds), *GenStat Reference Manual 3: Procedure Library PL17,*, VSN International, Hemel Hempstead, UK, pp. 260–265.

Welham, S. J., Cullis, B. R., Gogel, B. J., Gilmour, A. R. and Thompson, R. (2004). Prediction in linear mixed models, *Australian and New Zealand Journal of Statistics* **46**: 325–347.

Wolfinger, R. D. (1996). Heterogeneous variance-covariance structures for repeated measures, *Journal of Agricultural, Biological, and Environmental Statistics* **1**: 362–389.

Wolfinger, R. and O'Connell, M. (1993). Generalized linear mixed models: A pseudo-likelihood approach, *Journal of Statistical Computation and Simulation* **48**: 233–243.

Yates, F. (1935). Complex experiments, *Journal of the Royal Statistical Society, Series B* **2**: 181–247.