



## **User Guide**

Release 4.3

## **Functional Specification**

A R Gilmour  
VSN International, United Kingdom

B J Gogel  
Statistics for Agricultural Experiments (STAGE), Australia

B R Cullis  
University of Wollongong, Australia

S J Welham  
[www.stats4biol.co.uk](http://www.stats4biol.co.uk)

S A Gezan  
VSN International, United Kingdom

R Thompson  
Rothamsted Research, United Kingdom

April 2026

---

# ASReml User Guide Release 4.3 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, S.A. Gezan 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

|               |  |
|---------------|--|
| A. R. Gilmour | <a href="mailto:arthur.gilmour@cargovale.com.au">arthur.gilmour@cargovale.com.au</a> |
| B. J. Gogel   | <a href="mailto:beverley@stagece.au">beverley@stagece.au</a>                         |
| B. R. Cullis  | <a href="mailto:bcullis@uow.edu.au">bcullis@uow.edu.au</a>                           |
| S. J. Welham  | <a href="mailto:stats4biol@gmail.com">stats4biol@gmail.com</a>                       |
| S. A. Gezan   | <a href="mailto:salvador.gezan@vsni.co.uk">salvador.gezan@vsni.co.uk</a>             |
| R. Thompson   | <a href="mailto:robin.thompson@rothamsted.ac.uk">robin.thompson@rothamsted.ac.uk</a> |

## Copyright Notice

Copyright 2026 © VSN International

All rights reserved.

Except as permitted under the Copyright Act 1968 (Commonwealth of Australia), no part of the publication may be reproduced by any process, electronic or otherwise, without specific written permission of the copyright owner. Neither may information be stored electronically in any form whatever without such permission.

## **Published by:**

VSN International Ltd,  
Amberside House,  
Wood Lane,  
Paradise Industrial Estate,  
Hemel Hempstead,  
HP2 4TP, UK

email: [info@vsni.co.uk](mailto:info@vsni.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., Gezan, S.A. and Thompson, R. (2026).  
**ASReml User Guide Release 4.3 Functional Specification**, VSN International Ltd, Hemel  
Hempstead, HP2 4TP, UK, [www.vsni.co.uk](http://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 for further computational developments and research on the analysis of mixed models. Release 4 of ASReml was first distributed in 2014. The major enhancement was the introduction of the functional specification of linear mixed models. Release 4.2 in 2021 incorporated major enhancements to the computing speed and the size of problems that can be handled. Further developments in release 4.3 primarily relate to alternative, faster methods for large analyses. These are highlighted in a second document ASReml Update: What's new in Release 4.3.

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 fitting 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 VSNi and distributed with ASReml. ASReml-R refers to the R language interface to a DLL of the core ASReml routines. ASReml-SA is sometimes used to refer to the primary 'stand-alone' program when making a distinction to the R language ASReml-R interface. 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 16 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. [Chapter 16](#) 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 specification 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 document “*ASReml Update: What's New in Release 4.3*” details the new features, improvements and updates in this release. You can find this document in the `docs` directory of your **ASReml** installation. The changes include a reorganization of some core routines ([Section 11.5.1](#)), which enables **ASReml 4.3** to run substantially faster.

The data sets and **ASReml** input used in this guide are available from 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.

## Acknowledgements

We gratefully acknowledge the Grains Research and Development Corporation of Australia for their financial support for our research related to earlier releases of ASReml. Brian Cullis and Arthur Gilmour wish to thank the NSW Department of Primary Industries, and the University of Wollongong, for providing a stimulating and exciting environment for applied biometrical research and consulting. We sincerely thank Ari Verbyla, Dave Butler and Alison Smith, for their contributions to ASReml. 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 originally 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 initial 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. We thank Damian Collins who contributed the `PREDICT !PLOT` material and wrote the section on GLMM of this manual. Greg Dutkowski has contributed to the extended pedigree options. Alison Kelly has helped with the review of the **XFA** models. We thank Martyn Byng, Giovanni Galli, Valerie Poupon and Darren Murray from the ASReml team for their contribution in testing the application and editing the documentation. Finally, we especially thank our close associates who continually test the enhancements.

# Contents

|   |            |
|---|------------|
| <b>Preface .....</b>  | <b>i</b>   |
| <b>Contents .....</b>   | <b>iv</b>  |
| <b>List of tables .....</b>                                       | <b>xii</b> |
| <b>List of figures .....</b>                                      | <b>xiv</b> |
| <b>1 Introduction .....</b>                                       | <b>1</b>   |
| 1.1 What ASReml can do .....                                      | 1          |
| 1.2 Installation .....  | 1          |
| 1.3 User Interface .....  | 1          |
| 1.3.1 ASReml-W .....  | 2          |
| 1.3.2 ConTEXT .....   | 2          |
| 1.4 How to use this guide .....                                   | 2          |
| 1.5 Getting assistance .....                                      | 3          |
| 1.6 Typographic conventions .....                                 | 3          |
| <b>2 Some theory .....</b>  | <b>4</b>   |
| 2.1 The general linear mixed model .....                          | 4          |
| 2.1.1 Sigma parameterization of the linear mixed model .....      | 4          |
| 2.1.2 Partitioning the fixed and random model terms .....         | 5          |
| 2.1.3 G structure for the random model terms .....                | 5          |
| 2.1.4 Partitioning the residual error term .....                  | 5          |
| 2.1.5 R structure for the residual error term .....               | 6          |
| 2.1.6 Gamma parameterization for the linear mixed model .....     | 6          |
| 2.1.7 Parameter types .....                                       | 7          |
| 2.1.8 Variance structures for the random model terms .....        | 7          |
| 2.1.9 Variance models for terms with several factors .....        | 7          |
| 2.1.10 Direct product structures .....                            | 8          |
| 2.1.11 Direct products in R structures .....                      | 8          |
| 2.1.12 Direct products in G structures .....                      | 9          |
| 2.1.13 Range of variance models for R and G structures .....      | 9          |
| 2.1.14 Combining variance models in R and G structures .....      | 10         |
| 2.2 Estimation .....  | 10         |
| 2.2.1 Estimation of the variance parameters .....                 | 10         |
| 2.2.2 Estimation/prediction of the fixed and random effects ..... | 12         |
| 2.2.3 Use of the gamma parameterization .....                     | 13         |
| 2.3 What are BLUPs? .....   | 13         |
| 2.4 Inference: Random effects .....                               | 14         |

|          |  |           |
|----------|--|-----------|
| 2.4.1    | Tests of hypotheses: variance parameters .....                             | 14        |
| 2.4.2    | Diagnostics.....   | 15        |
| 2.5      | Inference: Fixed effects .....   | 17        |
| 2.5.1    | Introduction .....   | 17        |
| 2.5.2    | Incremental and conditional Wald F Statistics .....                        | 17        |
| 2.5.3    | Kenward and Roger adjustments .....  | 20        |
| 2.5.4    | Approximate stratum variances .....  | 21        |
| <b>3</b> | <b>A guided tour.....</b>  | <b>22</b> |
| 3.1      | Introduction .....   | 22        |
| 3.2      | Nebraska Intrastate Nursery (NIN) field experiment.....                    | 22        |
| 3.3      | The ASReml data file.....  | 23        |
| 3.4      | The ASReml command file .....  | 25        |
| 3.4.1    | Generating a template .....  | 25        |
| 3.4.2    | The title line .....   | 26        |
| 3.4.3    | Reading the data .....   | 27        |
| 3.4.4    | The data file line .....   | 27        |
| 3.4.5    | Tabulation .....   | 27        |
| 3.4.6    | Specifying the terms in the mixed model .....                              | 28        |
| 3.4.7    | Variance structures .....  | 28        |
| 3.4.8    | Prediction .....   | 29        |
| 3.5      | Running the job .....  | 29        |
| 3.6      | Description of output files .....  | 29        |
| 3.6.1    | The .asr file.....   | 30        |
| 3.6.2    | The .sln file.....   | 31        |
| 3.6.3    | The .yht file.....   | 32        |
| 3.7      | Tabulation, predicted values and functions of the variance components..... | 32        |
| <b>4</b> | <b>Data file preparation.....</b>  | <b>34</b> |
| 4.1      | Introduction .....   | 34        |
| 4.2      | The data file .....  | 34        |
| 4.2.1    | Free format data files.....  | 34        |
| 4.2.2    | Fixed format data files .....  | 36        |
| 4.2.3    | Preparing data files in Excel .....  | 36        |
| 4.2.4    | Binary format data files.....  | 36        |
| <b>5</b> | <b>Command file: Reading the data .....</b>                                | <b>37</b> |
| 5.1      | Introduction .....   | 37        |
| 5.2      | Important rules .....  | 37        |
| 5.3      | Title line.....  | 38        |



|          |   |           |
|----------|---|-----------|
| 5.4      | Specifying and reading the data .....                             | 38        |
| 5.4.1    | Data field definition syntax.....                                 | 38        |
| 5.4.2    | Storage of alphabetic factor labels.....                          | 41        |
| 5.4.3    | Ordering factor levels .....                                      | 42        |
| 5.4.4    | Skipping input fields .....                                       | 42        |
| 5.5      | Transforming the data .....                                       | 42        |
| 5.5.1    | Transformation syntax.....  | 44        |
| 5.5.2    | QTL marker transformations.....                                   | 48        |
| 5.5.3    | Remarks concerning transformations .....                          | 50        |
| 5.5.4    | Special note on covariates .....                                  | 51        |
| 5.6      | Data file line.....   | 51        |
| 5.6.1    | Data line syntax.....   | 52        |
| 5.7      | Data file qualifiers .....  | 52        |
| 5.7.1    | Reading data records from multiple files.....                     | 56        |
| 5.8      | Job control qualifiers.....                                       | 56        |
| <b>6</b> | <b>Command file: Specifying the terms in the mixed model.....</b> | <b>75</b> |
| 6.1      | Introduction .....  | 75        |
| 6.2      | Specifying model formulae in ASReml.....                          | 75        |
| 6.2.1    | General rules.....  | 76        |
| 6.2.2    | Examples .....  | 79        |
| 6.3      | Fixed terms in the model .....                                    | 80        |
| 6.3.1    | Primary fixed terms .....   | 80        |
| 6.3.2    | Sparse fixed terms .....  | 80        |
| 6.4      | Random and residual terms in the variance component model.....    | 80        |
| 6.5      | Interactions and conditional factors .....                        | 81        |
| 6.5.1    | Interactions.....   | 81        |
| 6.5.2    | Expansions.....   | 81        |
| 6.5.3    | Conditional factors.....  | 82        |
| 6.5.4    | Associated Factors.....   | 82        |
| 6.6      | Alphabetic list of model design functions .....                   | 83        |
| 6.7      | Weights .....   | 87        |
| 6.8      | Generalized Linear (Mixed) Models .....                           | 87        |
| 6.8.1    | Generalized Linear Mixed Models .....                             | 91        |
| 6.9      | Missing values.....   | 92        |
| 6.9.1    | Missing values in the response.....                               | 92        |
| 6.9.2    | Missing values in the explanatory variables.....                  | 92        |
| 6.10     | Some technical details about model fitting in ASReml .....        | 92        |

|          |   |           |
|----------|---|-----------|
| 6.10.1   | Sparse versus dense.....  | 92        |
| 6.10.2   | Ordering of terms in ASReml.....  | 93        |
| 6.10.3   | Aliasing and singularities .....  | 93        |
| 6.10.4   | Examples of aliasing .....  | 94        |
| 6.11     | Wald F Statistics.....  | 94        |
| <b>7</b> | <b>Specifying the variance structures.....</b>  | <b>96</b> |
| 7.1      | Applying variance models to random terms .....  | 96        |
| 7.2      | Process to define a consolidated model term .....   | 97        |
| 7.2.1    | Modelling a single variance structure over several model terms .....  | 99        |
| 7.3      | Applying variance structures to the residual error term .....   | 101       |
| 7.3.1    | Special properties and rules in defining the residual error term .....  | 102       |
| 7.3.2    | Using <code>sat()</code> to specify the residual model term for data with sections ..   | 103       |
| 7.3.3    | Using <code>!VPGROUP</code> to constrain variance components .....  | 104       |
| 7.4      | Identifiability.....  | 104       |
| 7.5      | A sequence of variance structures for the NIN data.....   | 104       |
| 7.6      | Sigma versus gamma parameterization.....  | 109       |
| 7.6.1    | Which variance parameterization is used during estimation?.....   | 109       |
| 7.6.2    | Switching from the gamma to the sigma parameterization.....   | 109       |
| 7.7      | Variance model function qualifiers .....  | 111       |
| 7.7.1    | Parameter equality constraints <code>!=s</code> .....   | 112       |
| 7.7.2    | Ways to supply distances in one-dimensional metric-based models<br><code>!COORD V</code> .....  | 114       |
| 7.7.3    | Your own program <code>!F i</code> .....  | 114       |
| 7.7.4    | Parameter space constraints <code>!G s</code> .....   | 115       |
| 7.7.5    | Initial values <code>!INIT v</code> .....   | 116       |
| 7.7.6    | Parameter types <code>!T s</code> .....   | 116       |
| 7.7.7    | Equating variance structures <code>!USE t</code> .....  | 117       |
| 7.8      | Setting relationships among variance structure parameters.....  | 118       |
| 7.8.1    | Simple relationships among variance structure parameters.....   | 118       |
| 7.8.2    | Fitting linear relationships among variance structure parameters .....  | 120       |
| 7.9      | Ways to present initial values to ASReml .....  | 122       |
| 7.9.1    | Using templates to set parametric information associated with variance<br>structures using <code>.tsv</code> and <code>.msv</code> files..... | 123       |
| 7.9.2    | Using estimates from simpler models .....   | 125       |
| 7.10     | Default variance structures in ASReml .....   | 126       |
| 7.11     | Variance model functions available in ASReml.....   | 127       |
| 7.11.1   | Forming variance models from correlation models .....   | 127       |
| 7.11.2   | Non-singular variance matrices .....  | 128       |

|           |  |            |
|-----------|--|------------|
| 7.11.3    | Notes on the variance models .....                                     | 128        |
| 7.11.4    | Notes on Matérn.....   | 129        |
| 7.11.5    | Notes on power models.....   | 131        |
| 7.11.6    | Notes on Factor Analytic models .....                                  | 131        |
| 7.12      | Variance models available in ASReml .....                              | 133        |
| <b>8</b>  | <b>Command file: Multivariate analysis .....</b>                       | <b>140</b> |
| 8.1       | Introduction .....   | 140        |
| 8.1.1     | Repeated measures on rats .....  | 140        |
| 8.1.2     | Wether trial data .....  | 140        |
| 8.2       | Model specification .....  | 141        |
| 8.3       | Residual variance structures .....                                     | 141        |
| 8.3.1     | Specifying multivariate variance structures in ASReml .....            | 142        |
| <b>9</b>  | <b>Command file: Genetic analysis .....</b>                            | <b>143</b> |
| 9.1       | Introduction .....   | 143        |
| 9.2       | The command file .....   | 143        |
| 9.3       | The pedigree file.....   | 144        |
| 9.4       | Reading in the pedigree file .....                                     | 145        |
| 9.5       | Pedigree modification qualifiers .....                                 | 148        |
| 9.6       | Genetic groups .....   | 149        |
| 9.7       | Reading a user defined (inverse) relationship matrix .....             | 150        |
| 9.7.1     | Genetic groups in GIV matrices .....                                   | 152        |
| 9.7.2     | The example continued .....  | 153        |
| 9.7.3     | Dealing with singularities .....                                       | 153        |
| 9.7.4     | Saving the G inverse .....   | 154        |
| 9.7.5     | Forming the H matrix.....  | 154        |
| 9.7.6     | Keeping the GRM in memory .....  | 155        |
| 9.7.7     | Eigen-Model transformation !EIGTRANSFORM.....                          | 156        |
| 9.7.8     | Binary G matrices.....   | 157        |
| 9.8       | Factor effects with large Random Regression Models .....               | 160        |
| 9.9       | Genetic Trimming in MET Models with !KEEPGRM, sat () and !VGROUP ..... | 166        |
| <b>10</b> | <b>Tabulation of the data and prediction from the model.....</b>       | <b>169</b> |
| 10.1      | Introduction .....   | 169        |
| 10.2      | Tabulation .....   | 169        |
| 10.3      | Prediction .....   | 170        |
| 10.3.1    | Underlying principles .....  | 170        |
| 10.3.2    | PREDICT syntax.....  | 172        |
| 10.3.3    | PREDICT failure .....  | 178        |

|           |  |            |
|-----------|--|------------|
| 10.3.4    | Associated factors .....   | 178        |
| 10.3.5    | Complicated weighting with !PRESENT .....  | 182        |
| 10.3.6    | Estimate linear functions of predicted values .....                                  | 183        |
| 10.3.7    | Examples .....   | 184        |
| 10.3.8    | Prediction using two-way interaction effects .....                                   | 185        |
| <b>11</b> | <b>Command file: Running the job .....</b>   | <b>186</b> |
| 11.1      | Introduction .....   | 186        |
| 11.2      | The command line .....   | 186        |
| 11.2.1    | Normal run .....   | 186        |
| 11.2.2    | Processing a .pin file.....  | 187        |
| 11.2.3    | Forming a job template from a data file .....  | 187        |
| 11.3      | Command line options.....  | 187        |
| 11.3.1    | Prompt for arguments (A) .....   | 188        |
| 11.3.2    | Output control (B, !OUTFOLDER, !XML) .....   | 189        |
| 11.3.3    | Debug command line options (D, E) .....  | 190        |
| 11.3.4    | Graphics command line options (G, H, I, N, Q).....                                   | 190        |
| 11.3.5    | Job control command line options (C, F, O, R, Y).....                                | 192        |
| 11.3.6    | Workspace command line options (W) .....   | 192        |
| 11.3.7    | Examples Command Line .....  | 193        |
| 11.4      | Advanced processing arguments.....   | 193        |
| 11.4.1    | Standard use of arguments .....  | 193        |
| 11.4.2    | Prompting for input.....   | 194        |
| 11.4.3    | Paths and Loops .....  | 194        |
| 11.4.4    | Order of Substitution .....  | 198        |
| 11.5      | Software performance .....   | 198        |
| 11.5.1    | Timing process.....  | 198        |
| 11.5.2    | Slow processes .....   | 200        |
| <b>12</b> | <b>Command file: Merging data files .....</b>  | <b>202</b> |
| 12.1      | Introduction .....   | 202        |
| 12.2      | Merge Syntax .....   | 202        |
| <b>13</b> | <b>Functions of variance components .....</b>  | <b>204</b> |
| 13.1      | Introduction .....   | 204        |
| 13.2      | Syntax .....   | 204        |
| 13.2.1    | Functions of components .....  | 206        |
| 13.2.2    | Convert CORUH and XFA to US.....   | 208        |
| 13.2.3    | Correlation.....   | 208        |
| 13.2.4    | Convert variance matrix of variable to variance matrix of transformed variable ..... | 209        |

|           |  |            |
|-----------|--|------------|
| 13.2.5    | Write components .....   | 209        |
| 13.2.6    | A more detailed example .....  | 209        |
| 13.2.7    | Conversion of variance matrix variables to variance matrix of transformed variables..... | 211        |
| 13.3      | VPREDICT: pin file processing .....  | 213        |
| 13.4      | Examples .....   | 214        |
| <b>14</b> | <b>Description of output files .....</b>   | <b>215</b> |
| 14.1      | Introduction .....   | 215        |
| 14.2      | An example .....   | 216        |
| 14.3      | Key output files.....  | 216        |
| 14.3.1    | The .asr file.....   | 217        |
| 14.3.2    | The .sln file.....   | 219        |
| 14.3.3    | The .yht file.....   | 220        |
| 14.4      | Other ASReml output files .....  | 221        |
| 14.4.1    | The .aov file.....   | 221        |
| 14.4.2    | The .asl file.....   | 224        |
| 14.4.3    | The .srs file.....   | 224        |
| 14.4.4    | The .msv file.....   | 225        |
| 14.4.5    | The .pvc file.....   | 225        |
| 14.4.6    | The .pvs file.....   | 226        |
| 14.4.7    | The .res file.....   | 226        |
| 14.4.8    | The .rsv file.....   | 233        |
| 14.4.9    | The .tab file.....   | 233        |
| 14.4.10   | The .tsv file.....   | 234        |
| 14.4.11   | The .vpc and .vpv files.....   | 234        |
| 14.4.12   | The .vrb file.....   | 235        |
| 14.4.13   | The .vvp file.....   | 236        |
| 14.4.14   | The .wvr file.....   | 236        |
| 14.5      | ASReml output objects and where to find them .....                                       | 236        |
| <b>15</b> | <b>Error messages .....</b>  | <b>239</b> |
| 15.1      | Introduction .....   | 239        |
| 15.2      | Common problems .....  | 239        |
| 15.3      | Things to check in the .asr file.....  | 241        |
| 15.4      | An example .....   | 242        |
| 15.5      | Information, Warning and Error messages .....  | 247        |
| <b>16</b> | <b>Examples .....</b>  | <b>259</b> |
| 16.1      | Introduction .....   | 259        |

|       |  |            |
|-------|--|------------|
| 16.2  | Split plot design – Oats.....                            | 259        |
| 16.3  | Unbalanced nested design – Rats.....                     | 262        |
| 16.4  | Source of variability in unbalanced data – Volts.....    | 266        |
| 16.5  | Balanced repeated measures - Height.....                 | 268        |
| 16.6  | Spatial analysis of a field experiment – Barley.....     | 275        |
| 16.7  | Unreplicated early generation variety trial – Wheat..... | 281        |
| 16.8  | Multi-Environment Trials.....                            | 286        |
|       | 16.8.1 An early generation trial.....                    | 286        |
|       | 16.8.2 Meta analysis of trial means.....                 | 292        |
| 16.9  | The reduced animal model (RAM).....                      | 296        |
| 16.10 | Paired Case-Control study – Rice.....                    | 298        |
|       | 16.10.1A multivariate approach.....                      | 299        |
|       | 16.10.2Standard analysis.....                            | 302        |
|       | 16.10.3Interpretation of results.....                    | 306        |
| 16.11 | Balanced longitudinal data – Oranges.....                | 308        |
| 16.12 | Generalized Linear (Mixed) Models – Sheep.....           | 317        |
| 16.13 | Multivariate animal genetics data – Sheep.....           | 322        |
|       | 16.13.1Half-sib data analysis.....                       | 323        |
|       | 16.13.2Animal model.....                                 | 331        |
|       | <b>Bibliography .....</b>                                | <b>336</b> |
|       | <b>Index .....</b>                                       | <b>341</b> |

# List of tables

|  |     |
|--|-----|
| Table 3.1: Trial layout and allocation of varieties to plots in the NIN field trial .....  | 23  |
| Table 5.1: Field types .....   | 39  |
| Table 5.2: Forms of data transformation qualifiers .....   | 44  |
| Table 5.3: List of transformation qualifiers and their actions with examples .....   | 45  |
| Table 5.4: Examples of transformations.....  | 50  |
| Table 5.5: Qualifiers relating to data input and output.....   | 53  |
| Table 5.6: List of commonly used job control qualifiers .....  | 56  |
| Table 5.7: List of occasionally used job control qualifiers .....  | 59  |
| Table 5.8: List of rarely used job control qualifiers .....  | 64  |
| Table 5.9: List of very rarely used job control qualifiers.....  | 72  |
| Table 6.1: Summary of reserved words, operators and functions .....  | 77  |
| Table 6.2: Simple examples of model formulae .....   | 79  |
| Table 6.3: Alphabetic list of model design functions and descriptions .....  | 83  |
| Table 6.4: Link qualifiers and functions.....  | 88  |
| Table 6.5: GLM distribution qualifiers.....  | 88  |
| Table 6.6: Examples of aliasing in ASReml.....   | 94  |
| Table 7.1: List of common variance model functions .....   | 97  |
| Table 7.2: Building consolidated model terms in ASReml .....   | 98  |
| 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 ..... | 110 |
| Table 7.4: Variance model function qualifiers available in ASReml .....  | 112 |
| Table 7.5: Examples of constraining variance parameters in ASReml .....  | 113 |
| Table 7.6: Variance parameter space constraint options .....   | 115 |
| Table 7.7: Variance components parameter types .....   | 117 |
| Table 7.8: VCC directive examples.....   | 119 |
| Table 7.9: Legacy and new VCC directive examples .....   | 120 |
| Table 7.10: Details of the variance functions available in ASReml .....  | 133 |
| Table 9.1: List of pedigree file qualifiers.....   | 146 |
| Table 9.2: Pedigree modification qualifiers .....  | 149 |
| Table 9.3: Qualifiers for reading GIV/GRM files grouped by functionality .....   | 151 |
| Table 9.4: GRM inversion qualifiers .....  | 154 |
| Table 9.5: GRM/GIV file extensions.....  | 157 |
| Table 9.6: Main GRR line qualifiers .....  | 161 |
| Table 9.7: Additional GRR line qualifiers .....  | 161 |
| Table 9.8: Rarely used GRR line qualifiers .....   | 162 |

|   |     |
|---|-----|
| Table 10.1: TABULATE directive qualifiers .....   | 170 |
| Table 10.2: List of prediction qualifiers.....  | 174 |
| Table 10.3: List of predict plot options .....  | 177 |
| Table 10.4: Trials classified by region and location and their means .....  | 179 |
| Table 11.1: Command line options .....  | 188 |
| Table 11.2: Graphic display options.....  | 190 |
| Table 11.3: Graphic file content substrings.....  | 191 |
| Table 11.4: Graphic file type qualifiers.....   | 191 |
| Table 11.5: Job control command options .....   | 192 |
| Table 11.6: Job control command examples.....   | 193 |
| Table 11.7: The use of arguments in ASReml .....  | 194 |
| Table 11.8: High-level qualifiers.....  | 195 |
| Table 11.9: Model processing details comparison .....   | 200 |
| Table 12.1: List of MERGE qualifiers .....  | 203 |
| Table 14.1: Summary of ASReml output files .....  | 215 |
| Table 14.2: ASReml output objects and where to find them .....  | 237 |
| Table 15.1: Some information messages and comments.....   | 248 |
| Table 15.2: List of warning messages and likely meaning(s) .....  | 249 |
| Table 15.3: Alphabetical list of error messages and probable cause(s)/remedies .....  | 251 |
| Table 16.1: A split-plot field trial of oat varieties and nitrogen application .....  | 259 |
| Table 16.2: Rat data: AOV decomposition .....   | 263 |
| Table 16.3: REML log-likelihood ratio for the variance components in the voltage data .....   | 268 |
| Table 16.4: Summary of variance models fitted to the plant data .....   | 270 |
| Table 16.5: Summary of Wald F statistics for fixed effects for variance models fitted to the plant data .....   | 275 |
| Table 16.6: Field layout of Slate Hall Farm experiment.....   | 276 |
| Table 16.7: Summary of models for the Slate Hall data .....   | 278 |
| Table 16.8: Estimated variance parameters from bivariate analysis of bloodworm data.....  | 301 |
| Table 16.9: 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 ..... | 304 |
| Table 16.10: Equivalence of random effects in bivariate and univariate analyses .....   | 306 |
| Table 16.11: Orange data: AOV decomposition.....  | 312 |
| Table 16.12: Sequence of models fitted to the Orange data .....   | 313 |
| Table 16.13: REML estimates of a subset of the variance parameters for each trait (expressed as a ratio to their asymptotic standard error).....  | 324 |
| Table 16.14: Wald F statistics of the fixed effects for each trait for the genetic example .....  | 324 |
| Table 16.15: Variance models fitted for each part of the ASReml job in the analysis of the genetic example.....   | 326 |



# List of figures

|   |     |
|---|-----|
| Figure 5.1: Variogram in 4 sectors for Cashmore data.....   | 72  |
| Figure 14.1: Residual versus Fitted values .....  | 221 |
| Figure 14.2: Variogram of residuals .....   | 230 |
| Figure 14.3: Plot of residuals in field plan order .....  | 231 |
| Figure 14.4: Plot of the marginal means of the residuals .....  | 232 |
| Figure 14.5: Histogram of residuals .....   | 232 |
| Figure 16.1: Residual plot for the rat data .....   | 265 |
| Figure 16.2: Residual plot for the voltage data.....  | 267 |
| Figure 16.3: Trellis plot of the height for each of 14 plants.....  | 269 |
| Figure 16.4: Residual plots for the <code>EXP</code> variance model for the plant data .....  | 272 |
| Figure 16.5: Sample variogram of the residuals from the <code>AR1</code> × <code>AR1</code> model.....  | 278 |
| Figure 16.6: Sample variogram of the residuals from the <code>AR1</code> × <code>AR1</code> model for the Tullibigeal data .....  | 284 |
| Figure 16.7: Sample variogram of the residuals from the <code>AR1</code> × <code>AR1</code> + <code>pol(column, -1)</code> model for the Tullibigeal data<br>- - - Results from analysis of yield - - - ..... | 284 |
| Figure 16.8: Plot of first and second loadings for XFA2 model (part 4).....   | 292 |
| Figure 16.9: Scatterplot of specific variance and loadings for the wheat dataset .....  | 295 |
| Figure 16.10: Rice bloodworm data: Plot of square root of root weight for treated versus control .....  | 299 |
| Figure 16.11: BLUPs for treated for each variety plotted against BLUPs for control.....   | 302 |
| Figure 16.12: Estimated deviations from regression of treated on control for each variety plotted against estimate for control.....   | 307 |
| Figure 16.13: Estimated difference between control and treated for each variety plotted against estimate for control .....  | 308 |
| Figure 16.14: Trellis plot of trunk circumference for each tree.....  | 309 |
| Figure 16.15: Fitted cubic smoothing spline for Tree 1 .....  | 312 |
| Figure 16.16: Plot of fitted cubic smoothing spline for model 1 .....   | 314 |
| Figure 16.17: Trellis plot of trunk circumference for each tree at sample dates (adjusted for <i>season</i> effects), with fitted profiles across time and confidence intervals.....                          | 316 |
| Figure 16.18: Plot of the residuals from the nonlinear model of Pinheiro and Bates.....   | 317 |

# 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.

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. Further details about the licensing can be found on the ASReml knowledge base: <https://asreml.kb.vsnl.co.uk/knowledge-base/vsnl-new-licensing/>. If you require help with installation or licensing, please contact support at <https://vsnl.co.uk/support>.

## 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 database 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

## 1.4 How to use this guide

---

- 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 available for Windows are presented below.

### 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 Windows 64 bit platform only.

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 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 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. Variance modelling is a complex aspect of analysis. We demonstrate 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 of fixed 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 data merging facility. [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.

## 1.5 Getting assistance

The ASReml help accessible through ASReml-W can also be linked to ConTEXT or accessed directly (ASReml.chm).

Users with a license with VSNi should visit <https://vsni.co.uk/support> 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.

## 1.6 Typographic conventions

A hands-on approach is the best way to develop a working understanding of a 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 blue type for easy identification (and the rest in green).
- The continuation symbol (:) is used to indicate that some of the original code is omitted.

An example ASReml code box

```
blue 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.

Other conventions are as follows

- 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.
- Example code and ASReml output are in `this size and font`.
- `This font` is used for all other code.
- Keyboard key names appear in SMALLCAPS, for example, TAB, COMMA and ESC.

## 2 Some theory

### 2.1 The general linear mixed model

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

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

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

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

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

$$\text{var}(\mathbf{y}) = \mathbf{Z}\mathbf{G}(\boldsymbol{\sigma}_g)\mathbf{Z}^\top + \mathbf{R}_v(\boldsymbol{\sigma}_r) \quad (2.3)$$

which we will refer to as the *sigma parameterization* of the  $\mathbf{G}$  and  $\mathbf{R}$  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  $\mathbf{G}$  and  $\mathbf{R}_v$  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  $\mathbf{u}$ , and a residual error term  $\mathbf{e}$ .

The two random components of this model, namely  $\mathbf{u}$  and  $\mathbf{e}$ , are each assumed to be independent and identically distributed (IID) and to follow a normal distribution such that  $\mathbf{u} \sim N(\mathbf{0}, \sigma_u^2 \mathbf{I}_q)$  and  $\mathbf{e} \sim N(\mathbf{0}, \sigma_e^2 \mathbf{I}_n)$ . Hence the variance of  $\mathbf{y}$  has the form

$$\text{var}(\mathbf{y}) = \sigma_u^2 \mathbf{Z}\mathbf{Z}^\top + \sigma_e^2 \mathbf{I}_n \quad (2.4)$$

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

## 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  $\boldsymbol{\tau} = [\boldsymbol{\tau}_1^\top \dots \boldsymbol{\tau}_t^\top]^\top$  and  $\boldsymbol{u}$  can be partitioned as  $\boldsymbol{u} = [\boldsymbol{u}_1^\top \dots \boldsymbol{u}_b^\top]^\top$ , with  $\boldsymbol{X}$  and  $\boldsymbol{Z}$  partitioned conformably as  $\boldsymbol{X} = [\boldsymbol{X}_1 \dots \boldsymbol{X}_t]$  and  $\boldsymbol{Z} = [\boldsymbol{Z}_1 \dots \boldsymbol{Z}_b]$ .

## 2.1.3 G structure for the random model terms

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

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

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

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

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

and

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

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

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

## 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  $\mathbf{e}$  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. Then, for  $s$  sections we have  $\mathbf{e} = [\mathbf{e}_1^\top, \mathbf{e}_2^\top, \dots, \mathbf{e}_s^\top]^\top$  assuming that the data vector is ordered by section, and where  $\mathbf{e}_j$  represents the vector of errors for the  $j^{th}$  section.

### 2.1.5 R structure for the residual error term

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

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

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  $\mathbf{R}_v$ .

In many cases, the residual errors ( $\mathbf{e}$ ) 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 7.10](#) 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  $\text{var}(\mathbf{y})$ . In this parameterization both  $\mathbf{G}(\sigma_g)$  and  $\mathbf{R}_v(\sigma_r)$  are variance matrices and the variance structure parameters in  $\sigma_g$  and  $\sigma_r$  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

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

for  $\gamma_{g_i}$  being the ratio of the variance component for the random term  $\mathbf{u}_i$  relative to error variance, that is,  $\gamma_{g_i} = \sigma_{u_i}^2 / \sigma_e^2$ . In this case **ASReml** calculated a simple estimate of  $\sigma_e^2$  and initial values for the iterative process were specified in terms of the ratios  $\gamma_{g_i}$  rather than in terms of the variance components  $\sigma_{u_i}^2$ . It was often easier to specify initial values in terms of these ratios rather than the variance components which is why this approach was adopted. Where  $\mathbf{R}_v(\sigma_r)$  can be written as a

## 2.1 The general linear mixed model

scaled correlation matrix, that is,  $\mathbf{R}_v(\sigma_r) = \sigma_e^2 \mathbf{R}_c(\gamma_r)$ , this suggests the alternative specification of (2.2)

$$\begin{bmatrix} \mathbf{u} \\ \mathbf{e} \end{bmatrix} \sim N \left( \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \end{bmatrix}, \sigma_e^2 \begin{bmatrix} \mathbf{G}(\gamma_g) & \mathbf{0} \\ \mathbf{0} & \mathbf{R}_c(\gamma_r) \end{bmatrix} \right) \quad (2.7)$$

where  $\gamma_g$  and  $\gamma_r$  represent the variance structure parameters associated with scaled (by  $\sigma_e^2$ ) variance matrices. In this case

$$\text{var}(\mathbf{y}) = \sigma_e^2 \left( \mathbf{ZG}(\gamma_g)\mathbf{Z}^\top + \mathbf{R}_c(\gamma_r) \right), \quad (2.8)$$

which we will refer to as the *gamma parameterization*, and the individual variance structure parameters in  $\gamma_g$  and  $\gamma_r$  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  $\sigma_g$  and  $\sigma_r$  and each gamma in  $\gamma_g$  and  $\gamma_r$  has a parameter type, for example, variance components, variance component ratios, autocorrelation parameters, factor loadings. Furthermore, the parameters in  $\sigma_g$ ,  $\sigma_r$ ,  $\gamma_g$  and  $\gamma_r$  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  $\mathbf{u}_i$  in  $\mathbf{u}$  define the random effects and associated design matrices,  $\mathbf{Z}_i \in \mathbf{Z}$ , but additional information is required before the model can be fitted.

This extra step involves defining the  $\mathbf{G}$  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  $\mathbf{G}_i$ . This produces a consolidated model term that simultaneously defines both the design matrix ( $\mathbf{Z}_i$ ) and variance model ( $\mathbf{G}_i$ ). 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, *i.e.* the levels of B cycling fastest

$$(\mathbf{ab}) = (ab_{11}, ab_{12}, \dots, ab_{1n}, ab_{21}, ab_{22}, \dots, ab_{2n}, \dots, ab_{m1}, ab_{m2}, \dots, ab_{mn})^\top$$

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

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



## 2.1 The general linear mixed model

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

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

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

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

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

$$\text{var}(\mathbf{A}) = \begin{bmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & \textcolor{magenta}{A}_{23} \\ A_{31} & A_{32} & A_{33} \end{bmatrix}$$

and

$$\text{var}(\mathbf{B}) = \begin{bmatrix} B_{11} & \textcolor{blue}{B}_{12} \\ B_{21} & B_{22} \end{bmatrix}$$

then

$$\text{cov}(\textcolor{magenta}{ab}_{21}, \textcolor{blue}{ab}_{32}) = \textcolor{magenta}{A}_{23} \times \textcolor{blue}{B}_{12}$$

### 2.1.10 Direct product structures

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

$$\begin{aligned} \text{var}((\mathbf{ab})) &= \mathbf{A} \otimes \mathbf{B} \\ &= \begin{bmatrix} \mathbf{A}_{11}\mathbf{B} & \cdots & \mathbf{A}_{1p}\mathbf{B} \\ \vdots & \ddots & \vdots \\ \mathbf{A}_{m1}\mathbf{B} & \ddots & \mathbf{A}_{mp}\mathbf{B} \end{bmatrix} \end{aligned}$$

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  $\mathbf{e}$  was from a field experiment laid out in a rectangular array of  $r$  rows by  $c$  columns, we could arrange the

residuals as a matrix and might consider that they were autocorrelated within rows and columns. Writing the residuals as a vector in field order, that is, by sorting the residuals rows within columns (plots within blocks) the variance of the residuals might then be

$$\sigma_e^2 \Sigma_c (\rho_c) \otimes \Sigma_r (\rho_r)$$

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

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

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

$$I_n \otimes \Sigma$$

where  $\Sigma^{(n_t \times n_t)}$  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  $\mathbf{u}$  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

$$\Sigma \otimes I_g$$

where  $\Sigma$  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  $\tau$  and prediction of  $u$  (although the latter may not always be of interest) for given  $\sigma_g$  and  $\sigma_r$ . 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

$$\mathbf{y} \sim N(\mathbf{X}\tau, \mathbf{H}) \quad (2.10)$$

where  $\mathbf{H} = \mathbf{ZG}(\sigma_g)\mathbf{Z}^T + \mathbf{R}_v(\sigma_r)$ . REML does not use (2.10) for estimation of variance parameters, but rather uses a distribution free of  $\tau$ , essentially based on error contrasts or *residuals*. The derivation given below is presented in Verbyla (1990).

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

$$\mathbf{L}_1^\top \mathbf{X} = \mathbf{I}_p, \mathbf{L}_2^\top \mathbf{X} = \mathbf{0}$$

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

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

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

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

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

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

$$\begin{aligned} \ell_R &= -0.5 (\log \det \mathbf{L}_2^\top \mathbf{H}^{-1} \mathbf{L}_2 + \mathbf{y}_2^\top (\mathbf{L}_2^\top \mathbf{H} \mathbf{L}_2)^{-1} \mathbf{y}_2) \\ &= -0.5 (\log \det \mathbf{X}^\top \mathbf{H}^{-1} \mathbf{X} + \log \det \mathbf{H} + \mathbf{y}^\top \mathbf{P} \mathbf{y}_2) \end{aligned} \quad (2.11)$$

where

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

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

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

$$\ell_R = -\frac{1}{2} (\log \det \mathbf{C} + \log \det \mathbf{R}_v + \log \det \mathbf{G} + \mathbf{y}^\top \mathbf{P} \mathbf{y}) \quad (2.12)$$

We can also write

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

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

$$U(\kappa_i) = \partial \ell_R / \partial \kappa_i = -\frac{1}{2} [\text{tr}(\mathbf{P} \mathbf{H}_i) - \mathbf{y}^\top \mathbf{P} \mathbf{H}_i \mathbf{P} \mathbf{y}] \quad (2.13)$$

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

The elements of the observed information matrix are

$$\begin{aligned} -\frac{\partial^2 \ell_R}{\partial \kappa_i \partial \kappa_j} &= \frac{1}{2} \text{tr}(\mathbf{P} \mathbf{H}_{ij}) - \frac{1}{2} \text{tr}(\mathbf{P} \mathbf{H}_i \mathbf{P} \mathbf{H}_j) \\ &\quad + \mathbf{y}^\top \mathbf{P} \mathbf{H}_i \mathbf{P} \mathbf{H}_j \mathbf{P} \mathbf{y} - \frac{1}{2} \mathbf{y}^\top \mathbf{P} \mathbf{H}_{ij} \mathbf{P} \mathbf{y} \end{aligned} \quad (2.14)$$

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

The elements of the expected information matrix are

$$E\left(-\frac{\partial^2 \ell_R}{\partial \kappa_i \partial \kappa_j}\right) = \frac{1}{2} \text{tr}(\mathbf{P}\mathbf{H}_i\mathbf{P}\mathbf{H}_j). \quad (2.15)$$

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

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

where  $\mathbf{U}(\boldsymbol{\kappa}^{(0)})$  is the score vector (2.13) and  $\mathbf{I}(\boldsymbol{\kappa}^0, \boldsymbol{\kappa}^0)$  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; Gilmour 2019). The matrix denoted by  $I_A$  is obtained by averaging (2.14) and (2.15) and approximating  $\mathbf{y}^\top \mathbf{P}\mathbf{H}_{ij}\mathbf{P}\mathbf{y}$  by its expectation,  $\text{tr}(\mathbf{P}\mathbf{H}_{ij})$  in those cases when  $\mathbf{H}_{ij} \neq 0$ . For variance components models (that is those linear with respect to variances in  $\mathbf{H}$ ), the terms in  $I_A$  are exact averages of those in (2.14) and (2.15). The basic idea is to use  $I_A(\kappa_i, \kappa_j)$  in place of the expected information matrix in (2.16) to update  $\boldsymbol{\kappa}$ .

The elements of  $I_A$  are

$$I_A(\kappa_i, \kappa_j) = \frac{1}{2} \mathbf{y}^\top \mathbf{P}\mathbf{H}_i\mathbf{P}\mathbf{H}_j\mathbf{P}\mathbf{y}. \quad (2.17)$$

The  $I_A$  matrix is the (scaled) residual sums of squares and products matrix of

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

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

$$\begin{aligned} \mathbf{y}_i &= \mathbf{H}_i\mathbf{P}\mathbf{y} \\ &= \mathbf{H}_i\mathbf{R}_v^{-1}\tilde{\mathbf{e}} \\ &= \mathbf{R}_{v_i}\mathbf{R}_v^{-1}\tilde{\mathbf{e}}, \kappa_i \in \sigma_r \\ &= \mathbf{Z}\mathbf{G}_i\mathbf{G}^{-1}\tilde{\mathbf{u}}, \kappa_i \in \sigma_u \end{aligned}$$

where  $\tilde{\mathbf{e}} = \mathbf{y} - \mathbf{X}\hat{\boldsymbol{\tau}} - \mathbf{Z}\tilde{\mathbf{u}}$ ,  $\hat{\boldsymbol{\tau}}$  and  $\tilde{\mathbf{u}}$  are solutions to (2.18). 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  $\mathbf{u}$  the objective function

$$\log f_Y(\mathbf{y} | \mathbf{u}; \boldsymbol{\tau}, \mathbf{R}_v) + \log f_U(\mathbf{u}; \mathbf{G})$$

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

## 2.3 What are BLUPs?

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

$$\begin{bmatrix} \boldsymbol{X}^\top \boldsymbol{R}_v^{-1} \boldsymbol{X} & \boldsymbol{X}^\top \boldsymbol{R}_v^{-1} \boldsymbol{Z} \\ \boldsymbol{Z}^\top \boldsymbol{R}_v^{-1} \boldsymbol{X} & \boldsymbol{Z}^\top \boldsymbol{R}_v^{-1} \boldsymbol{Z} + \boldsymbol{G}^{-1} \end{bmatrix} \begin{bmatrix} \hat{\boldsymbol{\tau}} \\ \tilde{\boldsymbol{u}} \end{bmatrix} = \begin{bmatrix} \boldsymbol{X}^\top \boldsymbol{R}_v^{-1} \boldsymbol{y} \\ \boldsymbol{Z}^\top \boldsymbol{R}_v^{-1} \boldsymbol{y} \end{bmatrix} \quad (2.18)$$

These can be written as

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

where

$$\boldsymbol{C} = \boldsymbol{W}^\top \boldsymbol{R}_v^{-1} \boldsymbol{W} + \boldsymbol{G}^*, \boldsymbol{\beta} = [\boldsymbol{\tau}^\top \boldsymbol{u}^\top]^\top$$

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,  $\sigma_e^2$  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  $\boldsymbol{y} = \boldsymbol{X}\boldsymbol{\tau} + \boldsymbol{Z}\boldsymbol{u} + \boldsymbol{e}$  where  $\boldsymbol{X} = \mathbf{1}_b \otimes \mathbf{1}_r$  is the design matrix for  $\boldsymbol{\tau}$  (the overall mean),  $\boldsymbol{Z} = \boldsymbol{I}_b \otimes \mathbf{1}_r$  is the design matrix for the  $b$  (random) treatment effects  $u_i$  and  $\boldsymbol{e}$  is the error vector. Assuming that the treatment effects are random implies that  $\boldsymbol{u} \sim N(\boldsymbol{A}\boldsymbol{\psi}, \sigma_b^2 \boldsymbol{I}_b)$ , for some design matrix  $\boldsymbol{A}$  and parameter vector  $\boldsymbol{\psi}$ . It can be shown that

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

where  $\bar{\mathbf{y}}$  is the vector of treatment means,  $\bar{y} \cdot \cdot$  is the grand mean. The differences of the treatment means and the grand mean are the estimates of treatment effects if treatment effects are fixed. The **BLUP** is therefore a weighted mean of the data-based estimate and the ‘prior’ mean  $A\psi$ . If  $\psi = \mathbf{0}$ , the **BLUP** in (2.19) becomes

$$\tilde{\mathbf{u}} = \frac{r\sigma_b^2}{r\sigma_b^2 + \sigma^2} (\bar{\mathbf{y}} - \mathbf{1}\bar{y} \cdot \cdot) \quad (2.20)$$

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

## 2.4 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  $\ell_{R2}$  is the **REML** log-likelihood of the more general model and  $\ell_{R1}$  is the **REML** log-likelihood of the restricted model (that is, the **REML** log-likelihood under the null hypothesis), then the **REMLRT** is given by

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

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 a hypothesis with the parameter on the boundary of the parameter space. It can be shown that for a single variance component, the theoretical asymptotic distribution of the **REMLRT** is a mixture of  $\chi^2$

variates, where the mixing probabilities are 0.5, one with 0 degrees of freedom (spike at 0) and the other with 1 degree of freedom. The approximate P-value for the **REMLRT** statistic ( $D$ ), is  $0.5(1 - \Pr(\chi_1^2 \leq d))$  where  $d$  is the observed value of  $D$ . This has a 5% critical value of 2.71 in contrast to the 3.84 critical value for a  $\chi^2$  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  $\chi^2$  variates from 0 to  $k$  degrees of freedom. See Self and Liang (1987) for details.

Tests concerning variance components in generally balanced designs, such as the balanced one-way classification, can be derived from the usual analysis of variance.

It can be shown that the **REMLRT** for a variance component being zero is a monotone function of the F statistic for the associated term.

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

$$\begin{aligned} \text{AIC} &= -2\ell_{Ri} + 2t_i \\ \text{BIC} &= -2\ell_{Ri} + t_i \log v \end{aligned} \quad (2.22)$$

where  $t_i$  is the number of variance parameters in model  $i$  and  $v = n - p$  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  $e$  or  $u$ . Firstly we note that the **BLUP** of the residual vector is given by

$$\begin{aligned} \tilde{e} &= y - W\tilde{\beta} \\ &= R_v P y \end{aligned} \quad (2.23)$$

It follows that

$$\begin{aligned} E(\tilde{e}) &= \mathbf{0} \\ \text{var}(\tilde{e}) &= R_v - WC^{-1}W^T \end{aligned}$$

The matrix  $WC^{-1}W^T$  (under the sigma parameterization) is the so-called ‘extended hat’ matrix. **ASReml** includes the  $\sigma^2$  in the hat matrix under the gamma parameterization. It is the linear mixed effects model analogue of  $\sigma^2 X(X^T X)^{-1}X^T$  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



- $\mathbf{G}^{-1}\mathbf{u}$  and  $\mathbf{G}^{-1}\mathbf{u}/\text{diag}\sqrt{\mathbf{G}^{-1}-\mathbf{G}^{-1}\mathbf{C}^{ZZ}\mathbf{G}^{-1}}$  to the `.sln` file
- $\mathbf{R}_v^{-1}\mathbf{e}$  and  $\mathbf{R}_v^{-1}\mathbf{e}/\text{diag}\sqrt{\mathbf{R}_v^{-1}-\mathbf{R}_v^{-1}\mathbf{W}\mathbf{C}^{-1}\mathbf{W}^T\mathbf{R}_v^{-1}}$  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  $\psi$ s.

The variogram has been suggested as a useful diagnostic for assisting with the identification of appropriate variance models for spatial data (Cressie, 1991). Gilmour *et al.* (1997) demonstrate its usefulness for the identification of the sources of variation in the analysis of field experiments. If the elements of the data vector (and hence the residual vector) are indexed by a vector of spatial coordinates,  $\mathbf{s}^i, i = 1, \dots, n$ , then the ordinates of the sample variogram are given by

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

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 Figure 14.2) 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 angle  $\theta_{ij} (-180 < \theta_{ij} < 180)$  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) centered on an angle of  $(i - 1) * 180/s$  ( $i = 1, \dots, s$ ). 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 structure 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  $H_0 : \mathbf{L}\boldsymbol{\tau} = \mathbf{l}$  for given  $\mathbf{L}, r \times p$ , and  $\mathbf{l}, r \times 1$ , is given by

$$W = (\mathbf{L}\hat{\boldsymbol{\tau}} - \mathbf{l})^\top \{\mathbf{L}(\mathbf{X}^\top \mathbf{H}^{-1} \mathbf{X})^{-1} \mathbf{L}^\top\}^{-1} (\mathbf{L}\hat{\boldsymbol{\tau}} - \mathbf{l}) \quad (2.24)$$

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 (1997) 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  $\mathbf{l} = \mathbf{0}$ , 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  $\mathbf{L}$  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

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

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

$$R(1)$$

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

where the  $R(\cdot)$  operator denotes the 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,  $R(B|1, A)$  represents the difference between the reduction in sums of squares between the so-called maximal ‘model’

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

and

$$y \sim 1 + A$$

Implicit in these calculations is that

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

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

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

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

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

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

the terms **A**, **B**, **C**, **A.B**, **A.C** and **B.C** are marginal to **A.B.C**. Nelder (1977, 1994) argues that meaningful and interesting tests for terms in such models can only be conducted for those tests which respect marginality relations. This philosophy underpins the following description of the second Wald statistic available in **ASReml**, the so-called ‘conditional’ Wald statistic. This method is invoked by 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.

## 2.5 Inference: Fixed effects

As a simple example, for the three-way factorial model the conditional Wald statistics would be computed as

| term  | sums of squares  | M code |
|-------|--|--------|
| I     | $R(I)$   |        |
| A     | $R(A   I, B, C, B.C) = R(I, A, B, C, B.C) - R(I, B, C, B.C)$   | A      |
| B     | $R(B   I, A, C, A.C) = R(I, A, B, C, A.C) - R(I, A, C, A.C)$   | A      |
| C     | $R(C   I, A, B, A.B) = R(I, A, B, C, A.B) - R(I, A, B, A.B)$   | A      |
| A.B   | $R(A.B   I, A, B, C, A.C, B.C) = R(I, A, B, C, A, B, A.C, B.C) - R(I, A, B, C, A.C, B.C)$                    | B      |
| A.C   | $R(A.C   I, A, B, C, A.B, B.C) = R(I, A, B, C, A, B, A.C, B.C) - R(I, A, B, C, A.B, B.C)$                    | B      |
| B.C   | $R(B.C   I, A, B, C, A.B, A.C) = R(I, A, B, C, A, B, A.C, B.C) - R(I, A, B, C, A.B, A.C)$                    | B      |
| A.B.C | $R(A.B.C   I, A, B, C, A.B, A.C, B.C) = R(I, A, B, C, A, B, A.C, B.C, A.B.C) - R(I, A, B, C, A.B, A.C, B.C)$ | C      |

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

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

The preceding table includes a 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 preceding M code letter, are marginal to at least one term in a higher group, and so forth. For example, in the table, model term **A.B** has M code B because it is marginal to model term **A.B.C** and model term **A** has M code A because it is marginal to **A.B**, **A.C** and **A.B.C**. Model term  $\mu$  (M code .) is a special case in that 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

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

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

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

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 ( $\nu_{1i}$ ) and the incremental Wald F statistic for each term. To this is added the conditional Wald F statistic and the **M** code if **!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 (Table 5.8) 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  $\nu_{2i}$ , Kenward and Roger, 1997) and a probability value if these can be computed. They will be for the conditional Wald F statistic if it is reported. The **!DDF i** (see Table 5.6) 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 this example, 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  $\hat{\mathbf{t}}$ . They argued that it is useful to consider an improved estimator of the variance matrix of  $\hat{\mathbf{t}}$  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  $\hat{\mathbf{t}}$  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  $\sigma_H^2 = 1$ ) where  $\mathbf{G} = \bigotimes_{j=1}^q \gamma_j \mathbf{I}_{b_j}$ , it is often possible to consider a natural ordering of the variance component parameters including  $\sigma^2$ . Based on an idea due to Thompson (1980), ASReml computes approximate stratum degrees of freedom and stratum variances by a modified Cholesky diagonalisation of the average information matrix. That is, if  $\mathbf{F}$  is the average information matrix for  $\boldsymbol{\sigma}$ , let  $\mathbf{U}$  be an upper triangular matrix such that  $\mathbf{F} = \mathbf{U}^\top \mathbf{U}$ . We define

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

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

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

Thence

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

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

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

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 database 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 User [Interface](#).

### 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

### 3.3 The ASReml data file

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

### 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 20 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.



### 3.3 The ASReml data file

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

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  $\times$  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** can automatically augment spatial data (see `!ROWFACTOR, !COLUMNFACTOR`).

```
variety id pid raw rep nloc yield lat long row column
LANCER 1 NA NA 1 4 NA 4.3 1.2 1 1
LANCER 1 NA NA 1 4 NA 4.3 2.4 2 1
LANCER 1 NA NA 1 4 NA 4.3 3.6 3 1
LANCER 1 NA NA 1 4 NA 4.3 4.8 4 1
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
LANCER 1 NA NA 1 4 NA 4.3 18 15 1
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
:
```

optional field labels  
file augmented by missing  
values for first 15 plots (and  
3 buffer plots on column 4, 6  
and 9)) and variety coded  
LANCER to complete the  
22 $\times$ 11 array  
:

original data  
:

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 including variance structures as 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                          |
|                                       | column 11                       |
| data field definition                 | nin89.asd !SKIP 1               |
| data file name and qualifiers         | tabulate yield ~ variety        |
| tabulate statement                    | yield ~ mu variety !r idv(repl) |
| linear mixed model definition         | residual idv(units)             |
| residual variance model specification | PREDICT variety                 |
| PREDICT statement                     |                                 |

### 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 has 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.

### 3.4 The ASReml command file

---

- 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 an 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 1 !RENAME !ARGS
Title: nin89. !DOPART $1
#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.

The `!A` after `variety` tells **ASReml** that the first field is an alphanumeric factor. A number (greater than 1, or `*`) after variable names, as shown for `repl`, `row` and `column`, tells **ASReml** that the variable defines a numeric factor with levels coded 1:<maximum value>. So, the fifth field, labelled `repl`, should contain numbers 1:4 for levels. Similarly for `row` and `column`. The other fields include variates (`yield`) and various other variables.

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

### 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 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!SKIP1
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/functional*; we suggest copying the data file to your 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\ASReml43` 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](#).

## 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`.

## 3.6 Description of output files

This file contains a summary of the data, the iteration sequence, estimates of the variance parameters and 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 (4.3), its build date (in square brackets) and the title of the job. The second line gives the build platform (Windows x64), the workspace allocated (2.0 Gbyte), the basename for the output files (`nin89`), and the run date and time. After a license report, the folder where the job resides is indicated. 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 4.3ni [14 Nov 2025] NIN Alliance Trial 1989      job heading
Windows x64      2.0 Gbyte  nin89_SAG  01 Dec 2025 18:11:36.534
* Licensed to: valid
* Your ASReml license expires in 31 days *
*****
* Contact support@asreml.co.uk for licensing and support      *
***** ARG *
Folder: C:\Users\ASReml_SA_Manual\NIN
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
13 idv(repl)    4

Notice: Specify !SIGMAP to allow the Sigma parameterization
Forming      61 equations:      57 dense.
Initial updates will be shrunk by factor      0.316
* This job uses all of the 4 processor threads. *
Notice: 1 singularities detected in design matrix.
1 LogL= -454.807      S2= 50.329      168 df      0.1000
2 LogL= -454.663      S2= 50.120      168 df      0.1173
3 LogL= -454.532      S2= 49.868      168 df      0.1463
```

### 3.6 Description of output files

```

4 LogL= -454.472      S2= 49.637      168 df    0.1866
5 LogL= -454.469      S2= 49.585      168 df    0.1986
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                                parameter estimates
idv(repl)           IDV_V      4      0.199323      9.88291      1.12      0 P
units              SCA_V      224      1.00000      49.5824      9.08      0 P

                                testing fixed effects
                                Wald F statistics
Source of Variation      NumDF      DenDF      F-inc      P-inc
12 mu                    1          3.0      242.06      <.001
1 variety                55         165.0      0.88      0.712
Notice: The DenDF values are calculated ignoring fixed/boundary/singular
variance parameters using algebraic derivatives.
13 idv(repl)                    4 effects fitted
* This job used at least .2 of the 2.0 Gbyte of primary workspace. *
Finished:    01 Dec 2025 18:11:36.965    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    | NE87615  | -2.875  | 4.979    |
| variety    | NE87619  | 2.700   | 4.979    |
| variety    | NE87627  | -5.337  | 4.979    |
| mu         | 1        | 28.56   | 3.856    |
| idv(repl)  | 1        | 1.880   | 1.755    |
| idv(repl)  | 2        | 2.843   | 1.755    |
| idv(repl)  | 3        | -0.8713 | 1.755    |
| idv(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 [2.4.2 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 |
| :      |        |          |       |
| 220    | 17.386 | 2.914    | 13.01 |
| 221    | 21.148 | 8.502    | 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
```

```
01 Dec 2025 18:11:36
```

```
Simple tabulation of yield
```

| variety  | Mean  |
|----------|-------|
| 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 |
| :        |       |

### 3.7 Tabulation, predicted values and functions of the variance components

---

|         |       |
|---------|-------|
| 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

01 Dec 2025 18:11:36  
nin89

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 -----

| 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 20 lines of the data file are as follows

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

### 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 Section

3.4.1) provided that the `!SKIP` qualifier (Table 5.5) 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 5.3)
- 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 interpreted 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 where 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 5.8. 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. Missing values are commonly represented in ASReml data files by `NA`, `*` or `..`. ASReml will also recognise empty fields as missing values in `.csv` 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 5.8), or the identifiers are whole numbers less than 9,999,999 and the `!RECODE` qualifier is specified (see Table 5.8).

## 5 Command file: Reading the data

### 5.1 Introduction

In the code box to the right is the **ASReml** command file `nin89ar.as` for a spatial analysis of the Nebraska Intrastate Nursery (NIN) field experiment introduced [Chapter 3](#). The lines that are highlighted in blue type relate to reading in the data. In this chapter we use this example to discuss reading in the data in detail.

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

```
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)
```

### 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
- 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 in 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 #
Alphanumeric
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 explicitly nominate how many are read with the **!READ** qualifier described in Table 5.8. No more than 10,000 variables may be read or formed.

Following are some details on the 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 following heading)
- Additional variables can be created by transformation qualifiers.

```
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)
```

### 5.4.1 Data field definition syntax

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

```
SPACE [ !CSKIP [c] ] label [field_type] [transformations]
```

- **SPACE** – is now optional

## 5.4 Specifying and reading the data

---

- `!CSKIP [c]` can be used to skip  $c$  (default 1) data fields
- **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 7.10) must not be used

- *field\_type* controls how a set of characters in the data file, referred to as a data field, is to be converted to a real number, and whether that real number is treated as a variate or as the level of a factor if specified in the linear model, see Table 3.1 [Table 5.1](#) below.
- *transformations* are described in Section [5.5](#).

Table 5.1: Field types

| qualifier   | action  |
|---|---|
| <b>Field_type for a variate</b>   |   |
| If no field type qualifier is given, the field is read as a real variable and treated as a variate  |   |
| <b>Field_types for a factor</b>   |   |
| 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. For the elements where the value of $n$ is required, a warning is printed if the nominated value 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. |   |
| <code>* or n</code>   | is used when the data field has values $1 \dots n$ directly coding for the factor unless the levels are to be labelled (see <code>!L</code> ), for example<br><br><code>row * # 1:12</code>   |
| <code>!L s</code>   | is used when the data field is numeric with values $1 \dots n$ and labels are to be assigned to the $n$ levels, for example<br><br><code>sex !L Male Female</code>  |
| <code>!A [n]</code>   | is required if the data field is alphanumeric, for example<br><br><code>location !A # names</code><br><br>Specify $n$ if there are more than 1000 classes over all factors indicating the expected number for this factor<br><br>If the alphanumeric field contains a SPACE or COMMA, it must be enclosed by quotes ('Willow Tree', "Yes, a Comment") |



## 5.4 Specifying and reading the data

| qualifier            | action  |
|----------------------|---|
| <code>!A !L s</code> | <p>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</p> <pre>SNP !A !L C:C C:T T:T</pre> <p>defines the levels over-riding the default, data dependent order.</p> <p>If there are many labels, they may be written over several lines by using a trailing COMMA to indicate continuation of the list. 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</p> <pre>Genotype !A !L MyNames.txt Genotype !A !L 'My Names.txt' Genotype !A !L 'MyNames'</pre> <p>Use a <code>!SKIP</code> 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 <code>genb6 gena1 gena5 genb2 genb4 gena3</code>. In this case</p> <pre>Genotype !A !L gena1 genb2 gena3 genb4</pre> <p>would result in the levels of Genotype being ordered <code>gena1 genb2 gena3 genb4 genb6 gena5</code>.</p> |
| <code>!I [n]</code>  | <p>is required if the data is numeric defining a factor but not <code>1..n</code>; <code>!I</code> must be followed by <code>n</code> if more than 1000 codes are present, for example</p> <pre>year !I # 1995 1996</pre>   |
| <code>!AS p</code>   | <p>is required if the data field has level names in common with a previous</p> <p><code>!A</code> or <code>!I</code> factor <code>p</code> and is to be coded identically, for example in a plant diallel experiment</p> <pre>Male !A 22 Female !AS Male # Integrated coding</pre>  |
| <code>!P</code>      | <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. For example</p> <pre>Animal !P # Coded according to pedigree file</pre>   |

### Field\_types for a group of m variates or factors

|                       |   |
|-----------------------|---|
| <code>!G m [I]</code> | <p>is used when <code>m</code> contiguous data fields comprise a set to be used together. The variables will be treated as factors if the second argument (<code>I</code>) setting the number of levels is present (it may be <code>*</code>). For example</p> <pre>:</pre> <pre>x1 x2 x3 x4 x5 y data.dat y ~ mu x1 x2 x3 x4 x5</pre> <p>is equivalent to</p> <pre>:</pre> <pre>X !G 5 y data.dat y ~ mu X</pre> |
|-----------------------|---|

### Delimited Field\_Types for variates

ASReml allows the reading delimited formats of dates and time to construct dates in Julian days and time in seconds and treated as variables

## 5.4 Specifying and reading the data

| qualifier | action   |
|-----------|--|
| ! DATE    | specifies the field has one of the date formats <i>dd/mm/yy</i> , <i>dd/mm/ccyy</i> , <i>dd-Mon-yy</i> , <i>dd-Mon-ccyy</i> and is to be converted into a Julian day where <i>dd</i> is a 1 or 2 digit day of the month, <i>mm</i> is a 1 or 2 digit month of the year, <i>Mon</i> is a three letter month name (Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec), <i>yy</i> is the year within the century (00 to 99), <i>cc</i> 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, <i>yy</i> of 0-32 is taken as 2000-2032, 33-99 taken as 1933-1999. |
| ! DMY     | specifies the field has one of the date formats <i>dd/mm/yy</i> or <i>dd/mm/ccyy</i> and is to be converted into a Julian day.   |
| ! MDY     | specifies the field has one of the date formats <i>mm/dd/yy</i> or <i>mm/dd/ccyy</i> and is to be converted into a Julian day.   |
| ! TIME    | specifies the field has the time format <i>hh:mm:ss</i> . and is to be converted to seconds past midnight where <i>hh</i> is hours (0 to 23), <i>mm</i> is minutes (0-59) and <i>ss</i> is seconds (0 to 59). The separator ':' must be present.   |

### 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 than 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 pre-specify 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 `!SORTed` 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. `!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. `!CSKIP f` is ignored when reading binary data.

## 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). Occasional users

## 5.5 Transforming the data

---

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 5.5). ASReml actually reads the data file *after* parsing the model line. It reads a line, converts the text into a vector of real values, 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... V10000 for transformation purposes. Values from the data file, ignoring any !SKIP 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 5.8. 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

## 5.5 Transforming the data

---

```
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  $\forall i$ , where  $i$  is a number, for variables to be referred to in a transformation.  $\forall i$  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, and they are presented in [Table 5.2](#).

Table 5.2: Forms of data transformation qualifiers

| qualifier   | action   |
|---|--|
| <code>!operator</code>                            | to perform an operation on the current field, for example to take absolute values we use <code>absY !ABS</code> .  |
| <code>!operator value</code>                      | to perform an operation involving an argument on the current field, for example, <code>logY !=Y !^0</code> copies Y and then takes logs.                 |
| <code>!operator <math>\forall</math> field</code> | to perform an operation on the current field using the data in another field, for example, <code>!-V2</code> to subtract field 2 from the current field. |
| <code>!<math>\forall</math> target</code>         | to reset the focus for subsequent transformations to field number <i>target</i> .  |
| <code>!TARGET target</code>                       | to reset the focus for subsequent transformations to the previously named field <i>target</i> .  |
| <code>!<math>\forall</math> target = value</code> | to set the target field to a particular value.   |

## 5.5 Transforming the data

| qualifier                        | action  |
|----------------------------------|---|
| <code>!V target = V field</code> | to overwrite the data in a target field by the data values of another field; a special case is when <i>field</i> is 0 instructing ASReml to put the record number into the <i>target</i> field. |

- *operator* is one of the symbols defined in Table 5.3.
- *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.
- *Vfield* 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 5.3: List of transformation qualifiers and their actions with examples

| qualifier                   | argument | action   | examples  |
|-----------------------------|----------|--|---|
| <code>!=</code>             | <i>v</i> | used to overwrite/create a variable with <i>v</i> . It usually implies the variable is not read (see examples on page 43). | <code>one !=1</code><br><code>quarter !=0.25</code>     |
| <code>!+, !-, !*, !/</code> | <i>v</i> | usual arithmetic meaning; note that, 0/0 gives 0 but <i>v</i> /0 gives a missing value where <i>v</i> is not 0.            | <code>yield !/10</code>                                 |
| <code>!^</code>             | <i>v</i> | raises the data (which must be positive) to the power <i>v</i> .   | <code>yield</code><br><code>SQRyld !=yield !^0.5</code> |
| <code>!^</code>             | 0        | takes natural logarithms of the data (which must be positive).   | <code>Yield</code><br><code>LNyld !=yield !^0</code>    |
| <code>!^</code>             | -1       | takes reciprocal of data (data must be positive).  | <code>yield</code><br><code>INVyld !=yield !^-1</code>  |

## 5.5 Transforming the data

| qualifier   | argument        | action  | examples  |
|---|-----------------|---|---|
| !>, !<, $v$<br>!<>, !==,<br>!<=, !>=                |                 | logical operators forming 1 if true, 0 if false.  | yield<br>high !=yield !>10                                |
| !ABS  |                 | takes absolute values - no argument required.   | yield<br>ABSyld !=yield !ABS                              |
| !ARCSIN   | $v$             | forms an <b>ArcSin</b> transformation using the sample size specified in the argument, a number or another field. In the side example, for two existing fields <i>Germ</i> and <i>Total</i> containing counts, we form the <b>ArcSin</b> for their ratio ( <i>ASG</i> ) by copying the <i>Germ</i> field and applying the <b>ArcSin</b> transformation using the <i>Total</i> field as sample size.                   | Germ<br>Total<br>ASG !=Germ !ARC Total                    |
| !COS, !SIN  | $s$             | takes cosine and sine of the data variable with period $s$ having default $2\pi$ ; omit $s$ if data is in radians, set $s$ to 360 if data is in degrees.  | Day<br>CosDay !=Day !COS 365                              |
| !D,<br>!D<>,<br>!D<,<br>!D<=,<br>!D>,<br>!D>=       | $v$             | !D[ $o$ ] $v$ discards records which have $v$ or 'missing value' in the field, subject to the logical operator $o$ .  | yield !DV<=0<br>yield !DV<1 !DV>100                       |
| !DV,<br>!DV<>,<br>!DV<,<br>!DV<=,<br>!DV>,<br>!DV>= | $v$             | !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<br>yield !DV<1<br>!DV>100<br>InitialWt !DV * |
| !DO   | $[n[i_t[i_v]]]$ | causes <b>ASReml</b> 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 !DOM  |
| !DOM  | $f$             | copies and converts additive marker covariables (-1, 0, 1) to dominance marker covariables (see !DOM).  | ChrAadd !G 10 MM!..<br>ChrAdom !DOM ChrAadd               |
| !ENDDO  |                 | terminates a !DO transformation block.  | See !DOM  |
| !EXP  |                 | takes antilog base $e$ - no argument required.  | Rate !EXP   |

## 5.5 Transforming the data

| qualifier  | argument     | action  | examples   |
|--|--------------|---|--|
| !Jddm,<br>!Jmmd<br>!Jyyd                                     |              | !Jddm converts a number representing a date in the form <i>ddmmccyy</i> , <i>ddmmyy</i> or <i>ddmm</i> into days. !Jmmd converts a date in the form <i>ccyymmdd</i> , <i>yyymmdd</i> or <i>mmdd</i> into days. !Jyyd converts a date in the form <i>ccyyddd</i> or <i>yyddd</i> 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 <i>cc</i> is omitted it is taken as 19 if <i>yy</i> > 32 and 20 if <i>yy</i> < 33, the date must be entirely numeric: characters such as / may not be present (but see !DATE). |  |
| !M, !M<>, <i>v</i><br>!M< !M<= <i>v</i><br>!M> !M>= <i>v</i> |              | !M <i>v</i> converts data values of <i>v</i> to missing; if !M is used after !A or !I, <i>v</i> should refer to the factor level rather than the value in the data file (see also Section 4.2).   | yield !M-9<br>yield !M<=0 !M>100   |
| !MAX,<br>!MIN,<br>!MOD                                       | <i>v</i>     | the maximum, minimum and modulus of the field values and the value <i>v</i> .   | yield !MAX 9   |
| !MM  | <i>s</i>     | assigns Haldane map positions ( <i>s</i> ) to marker variables and imputes missing values to the markers (see Section 5.5.2).   | ChrAadd !G 10 !MM 1 ...  |
| !NA  | <i>v</i>     | replaces any missing values in the variate with the value <i>v</i> . If <i>v</i> is another field, its value is copied.   | Rate !NA 0<br>WT !=Wt2 !NA Wt1   |
| !NORMAL  | <i>v</i>     | replaces the variate with normal random variables having variance <i>v</i> .  | Ndat !=0 !Normal 4.5<br>is equivalent to<br>Ndat !=Normal 4.5                            |
| !REPLACE   | <i>o n</i>   | replaces data values of <i>o</i> with <i>n</i> in the current variable.   | Rate !REPLACE -9 0   |
| !RESCALE   | <i>o s</i>   | rescales the column(s) in the current variable (!G group of variables) using $Y = (Y + o) * s$  | Rate !RESCALE -10 0.1  |
| !SEED  | <i>v</i>     | sets the seed for the random number generator.  | !SEED 848586   |
| SET  | <i>vlist</i> | for <i>vlist</i> , a list of <i>n</i> values, the data values 1 · · · <i>n</i> are replaced by the corresponding element from <i>vlist</i> ; data values that are < 1 or > <i>n</i> are replaced by zero. <i>vlist</i> may run over several lines provided each incomplete line ends with a COMMA, i.e., a COMMA is used as a continuation symbol.  | treat !L C A B<br>CvR !=treat,<br>!SET 1 -1 -1<br><br>group !=treat !SET 1,<br>2 2 3 3 4 |
| !SETN  | <i>v n</i>   | !SETN <i>v n</i> replaces data values 1 : <i>n</i> with normal random variables having variance <i>v</i> . Data values outside the range 1 · · · <i>n</i> are set to 0.   | Anorm !=A !SETN 2.5 10   |



## 5.5 Transforming the data

| qualifier | argument | action   | examples  |
|-----------|----------|--|---|
| !SETU     | $v$ $n$  | replaces data values $1 : n$ with uniform random variables having range $0 : v$ . Data values outside the range $1 \cdot \dots \cdot n$ are set to 0.  | <code>Aeff !=A !SETU 5 10</code>                            |
| !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. | <code>year 3 !SUB 66 67 68</code>                           |
| !TARGET   | $v$      | changes the focus of subsequent transformations to variable (field) $v$ . The example starts with $a$ and $b$ , replace $b$ with $(a+b)$ and then replaces $a$ with $\sqrt{a}$ .   | <code>A<br/>B !+A<br/>!TARGET A !^0.5</code>                |
| !UNIFORM  | $v$      | replaces the variate with uniform random variables having range $0:v$ .  | <code>Udat !=0 !UNIFORM 4.5</code>                          |
| !Vtarget  | $value$  | assigns $value$ to data field $target$ overwriting previous contents; subsequent transformation qualifiers will operate on data field $target$ .   | <code>... !V3=2.5</code>                                    |
|           | $Vfield$ | 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.   | <code>... !V10=V3<br/>... !V11=block<br/>... !V12=V0</code> |

### 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

$$\theta_{LR} = (1 - e^{-2(s_R - s_L)})/2.$$

## 5.5 Transforming the data

---

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 (see Table 6.3).

`!DO . . . !ENDDO` provides a mechanism to repeat transformations on a set of variables. All transformations 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
Ymean !=0. !DO 5 0 1 !+Y1 !ENDDO !/5 # Repeat 5 times, incrementing
                                         # just 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 the rest
Markers !G 12 !DO !D * !ENDDO # Delete records with missing marker values
```

The default arguments ( $n = 12$ ,  $i_t = 1$ , and  $i_v = 0$ .) 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).
```
- 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`.

Table 5.4: Examples of transformations

| ASReml code   | action   |
|---|--|
| <code>yield !M0</code>  | changes the zero entries in <code>yield</code> to missing values.  |
| <code>yield !^0</code>  | takes natural logarithms of the <code>yield</code> data.   |
| <code>score !-5</code>  | subtracts 5 from all values in <code>score</code> .  |
| <code>score !SET -0.5 1.5 2.5</code>                                  | replaces data values of 1, 2 and 3 with -0.5, 1.5 and 2.5, respectively.   |
| <code>score !SUB -0.5 1.5 2.5</code>                                  | replaces data values of -0.5, 1.5 and 2.5 with 1, 2 and 3, respectively;<br>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. |
| <code>block 8</code><br><code>variety 20</code><br><code>yield</code> | in the case where <ul style="list-style-type: none"> <li>• there are multiple units per plot</li> </ul>  |

## 5.6 Data file line

| ASReml code   | action   |
|---|--|
| <code>plot * !=variety !SEQ</code>  | <ul style="list-style-type: none"><li>contiguous plots have different treatments, and</li><li>the records are sorted units within plots within blocks</li></ul> <p>this code generates a <code>plot</code> factor assuming a new plot whenever the code in <code>V2</code> (<code>variety</code>) changes; whether this creates a variable or overwrites an input variable depends on whether any subsequent variables are input variables.</p>  |
| <code>Var 3</code><br><code>Nit 4</code><br><code>VxN 12 !=Var !-1 !*4 !+Nit</code> | <p>assuming <code>Var</code> is coded 1:3 and <code>Nit</code> is coded 1:4, this syntax could be used to create a new factor <code>VxN</code> with the 12 levels of the composite <code>Var</code> by <code>Nit</code> factor.</p>  |
| <code>YA !V98=YA !NA 0</code><br><code>YB !V99=YB !NA 0 !+V98 !D0</code>            | <p>will discard records where <b>both</b> <code>YA</code> and <code>YB</code> have missing values (assuming neither have zero as valid data). The first line sets the focus to variable 98, copies <code>YA</code> into <code>V98</code> and changes any <i>missing values</i> in <code>V98</code> to zero. The second line sets the focus to variable 99, copies <code>YB</code> into <code>V99</code> and changes any <i>missing values</i> in <code>V99</code> to zero. It then adds <code>V98</code> and discards the whole record if the result is zero, <i>i.e.</i> both <code>YA</code> and <code>YB</code> have missing values for that record. Variables 98 and 99 are not labelled and so are not retained for subsequent use in analysis.</p> |

### 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 0.142857 # in weeks`

## 5.6 Data file line

The purpose of the data file 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
yield ~ mu variety
residual idv(units)
```

### 5.6.1 Data line syntax

The data file line appears in the **ASReml** command file in the form

*datafile* [*qualifiers*]  
[*newlinequalifiers*]

- *datafile* is the path name of the file that contains the data; include the path in the filename if the file is not in the current ‘working’ folder; the data includes 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 5.5 **Error! Reference source not found.** and Table 5.6 for a list of data file related qualifiers,
  - The operation of **ASReml**, see Table 5.6 to Table 5.9 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.
- *newlinequalifier* is used to add additional qualifiers including the more complex model and job control qualifiers. Those having multiple arguments need to appear on separate lines between the *datafile* line and the model line. These include `!DEFINE`, `!GROUPFACTOR`, `!MBF`, `!SPL`, `!SUBGROUP`, `!SUBSET`, `!GINDEX` and `!MERGE`.
- The arguments to qualifiers are represented by the following symbols
  - *f* — a filename
  - *n* — an integer number, typically a count
  - *p* — a list of real numbers, typically in increasing order. It is generally true that if there are many numbers in a list they may be written over several lines by using a trailing COMMA to indicate continuation of the list
  - *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 5.5 lists the qualifiers relating to data input. Use the [Index](#) to check for examples or further discussion of these qualifiers.

Table 5.5: Qualifiers relating to data input and output

| qualifier  | action  |
|--|---|
| <b>frequently used data file qualifiers</b>  |   |
| <code>!SKIP <i>n</i></code>  | causes the first <i>n</i> records of the (non-binary) data file to be ignored.<br>Typically, these lines contain column headings for the data fields.   |
| <b>other data file qualifiers</b>  |   |
| <code>!COLUMNFACTOR <i>v</i></code><br><code>!COLFAC <i>v</i></code>                                 | is used in combination with <code>!ROWFACTOR</code> [and <code>!SECTION</code> ] to get <b>ASReml</b> to insert extra data records to complete the grid of plots defined by the <i>RowFactor</i> and the <i>ColumnFactor</i> for each <i>Section</i> so that a two- dimensional error structure can be defined (see <code>!SECTION</code> in Table 5.7).  |
| <code>!CSV</code>  | used to make consecutive commas imply a missing value; this is automatically set if the file name ends with <code>.csv</code> or <code>.CSV</code> (see Section 4.2).<br><b>Warning</b> This qualifier is ignored when reading binary data.   |
| <code>!DATAFILE <i>f</i></code>  | specifies the datafile name replacing the one obtained from the datafile line. It is required when different <code>!PATHS</code> (see <code>!DOPATH</code> in Table 11.8) of a job must read different files. The <code>!SKIP</code> qualifier, if specified, will be applied when reading the file.  |
| <code>!FILTER <i>v</i></code><br><code>[!SELECT <i>n</i>]</code><br><code>[!EXCLUDE <i>n</i>]</code> | enables a subset of the data to be analysed; <i>v</i> is the number or name of a data field. When reading data, the value in field <i>v</i> is checked after any transformations are performed. If <code>!SELECT</code> and <code>!EXCLUDE</code> are omitted, records with zero in field <i>v</i> are omitted from the analysis. If <code>!SELECT <i>n</i></code> is specified records with <i>n</i> in field <i>v</i> are retained and all other records are omitted. Conversely if <code>!EXCLUDE <i>n</i></code> is specified, records with <i>n</i> in field <i>v</i> are ignored. |
| <code>!FOLDER <i>s</i></code>  | specifies an alternative folder for <b>ASReml</b> 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<br><code>!FOLDER ../Data</code><br><code>data.asd !SKIP 1</code><br>is equivalent to<br><code>../Data/data.asd !SKIP 1</code>  |
| <code>!FORMAT <i>s</i></code>  | supplies a <b>FORTRAN</b> like <code>FORMAT</code> statement for reading fixed format files. A simple example is<br><code>!FORMAT (3I4, 5F6.2)</code><br>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<br><code>!FORMAT (4x, 3 (I4, f8.2) )</code><br>Field descriptors are   |

## 5.7 Data file qualifiers

| qualifier | action  |
|-----------|---|
|           | <ul style="list-style-type: none"> <li>• <math>rX</math> to skip <math>r</math> character positions</li> <li>• <math>rAw</math> to define <math>r</math> consecutive fields of <math>w</math> characters width</li> <li>• <math>rIw</math> to define <math>r</math> consecutive fields of <math>w</math> characters width, and</li> <li>• <math>rFw.d</math> to define <math>r</math> consecutive fields of <math>w</math> characters width; <math>d</math> indicates where to insert the decimal point if it is not explicitly present in the field where <math>r</math> is an optional repeat count.</li> </ul> |

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 is 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

| qualifier   | action   |
|---|--|
| <code>!MERGE <i>c f</i> [ <code>!SKIP <i>n</i></code> ] [ <code>!MATCH <i>a b</i></code> ]</code> | <p>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 (<i>f</i>). This <code>!MERGE</code> qualifier has been superseded by the much more powerful <code>MERGE</code> statement (see <a href="#">Chapter 12</a>). The effect is to open the named file (skip <i>n</i> lines) and then insert the columns from the new file into field positions starting at position <i>c</i>. If <code>!MATCH <i>a b</i></code> is specified, <b>ASReml</b> checks that the field <i>a</i> (<math>0 &lt; a &lt; c</math>) has the same value as field <i>b</i>. 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 <code>MERGE</code> file is kept for comparison with the next record.</p> <p>It is assumed that the lines in the <code>MERGE</code> 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 <code>MERGE</code> file.</p> <p>For example, assuming the field definitions define 10 fields,</p> <pre>primary.dat !SKIP 1 !MERGE 6 second.dat !SKIP 1 !MATCH 1 6</pre> <p>would obtain the first five fields from <code>primary.dat</code> and the next five from <code>second.dat</code>, checking that the first field in each file has the same value.</p> <p>Thus, each input record is obtained by combining information from each file, before any transformations are performed.</p> |
| <code>!READ <i>n</i></code>   | formally instructs <b>ASReml</b> to read <i>n</i> 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 <b>ASReml</b> attempts to read more fields than it needs to.  |
| <code>!RECODE</code>  | 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 <code>!SAVE</code> option but will be needed if formed in some other way (see <a href="#">Section 4.2</a> ).  |
| <code>!ROWFACTOR <i>v</i></code><br><code>!ROWFAC <i>v</i></code>                                 | is used in combination with <code>!COLUMNFACTOR</code> [and <code>!SECTION</code> ] to get <b>ASReml</b> to insert extra data records to complete the grid of plots defined by the <i>RowFactor</i> and the <i>ColumnFactor</i> for each <i>Section</i> so that a two- dimensional error structure can be defined (see <code>!SECTION</code> in <a href="#">Table 5.6</a> ).   |
| <code>!RREC [<i>n</i>]</code>   | causes <b>ASReml</b> to read <i>n</i> records or to read up to a data reading error if <i>n</i> 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 <code>.pvs</code> file). The argument ( <i>n</i> ) specifies the number of data records to be read. If not supplied, <b>ASReml</b> reads until a data reading error occurs, and then processes the data it has. Without this qualifier, <b>ASReml</b> aborts the job when it encounters a data error. See <code>!RSKIP</code> .   |
| <code>!RSKIP <i>n</i> [<i>s</i>]</code>   | <p>allows <b>ASReml</b> to skip lines at the heading of a file down to (and including) the <i>n</i>th instance of string <i>s</i>. For example, to read back the third set predicted values in a <code>.pvs</code> file, you would specify</p> <pre>!RREC !RSKIP 4 ' Ecode'</pre> <p>since the line containing the 4th instance of <code>' Ecode'</code> immediately precedes the predicted values. The <code>!RREC</code> qualifier means that <b>ASReml</b> will read until the end of the predict table. The keyword <code>Ecode</code> which occurs once at the beginning and then immediately before each block of data in the <code>.pvs</code> file is used to count the sections.</p>  |



### 5.7.1 Reading data records from multiple 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
- !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 5.6: List of commonly used job control qualifiers

| qualifier  | action   |
|--|--|
| !CONTINUE [ <i>f</i> ]<br>!MSV [ <i>f</i> ]<br>!TSV [ <i>f</i> ] | <p>These qualifiers are used to restart/resume iterations from the point reached in a previous run. The qualifier !CONTINUE [<i>f</i>] can alternately be set from the command line using the option letter C [<i>f</i>] (see Section 11.3).</p> <p>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 and is more easily edited.</p> <p>If <i>f</i> is not set, then ASReml looks for a .rsv file with the same name used for the output files, <i>i.e.</i> the .as name possibly augmented by job arguments. ASReml then scans this file for parameter values related to the current model, replacing the original initial values obtained by default or from the .as file with the .rsv values before iteration commences. If !CONTINUE 2 or !TSV are used then the</p> |

## 5.8 Job control qualifiers

| qualifier   | action   |
|---|--|
|   | <p><code>.tsv</code> file is used instead of the <code>.rsv</code> file. Similarly, if <code>!CONTINUE 3</code> or <code>!MSV</code> are used then the <code>.msv</code> file is used instead of the <code>.rsv</code> file.</p> <p>If <i>f</i> is a file name with the file extension <code>.rsv</code>, <code>.tsv</code>, or <code>.msv</code>, initial values will be updated using it. Otherwise, <b>ASReml</b> will check for the file <i>ff.rsv</i>, <i>f.tsv</i> or <i>f.msv</i> to use. If no matching file is found, <b>ASReml</b> will look for the default <code>.rsv</code> file to use. Some users may prefer, rather than specifying initial values in the model formulation, to generate a default <code>.tsv</code> file using <code>!MAXIT 0</code> and then edit the <code>.tsv</code> file with more appropriate values. If the model has changed and <code>!CONTINUE</code> is used, <b>ASReml</b> will pick up the values it recognises as being for the same terms from the <code>.rsv</code> file.</p> <p>Furthermore, <b>ASReml</b> will use estimates in the <code>.rsv</code> 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.</p>                      |
| <code>!DEFINE s t p</code><br>formerly <code>!CONTRAST</code> | <p>provides a convenient way to define how particular effects, ‘contrasts’ among treatment levels, are defined. <code>!DEFINE</code> lines occur as separate lines between the datafile line and the model line.</p> <ul style="list-style-type: none"> <li>• <i>s</i> is the name of the model term being defined.</li> <li>• <i>t</i> is the name of an existing factor.</li> <li>• <i>p</i> is the list of contrast coefficients.</li> </ul> <p>For example</p> <pre><code>!DEFINE LinN Nitrogen 3 1 -1 -3</code></pre> <p>defines <code>LinN</code> as a ‘contrast’ based on the 4 (implied by the length of the list) levels of factor <code>Nitrogen</code>. 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 the ‘contrast’ (check the <code>.ass</code> or <code>.sln</code> or <code>.tab</code> files). It may also be used on the implicit factor <code>Trait</code> in a multivariate analysis provided it implicitly identifies the number of levels of <code>Trait</code>; the number of traits is implied by the length of the list. Thus, if the analysis involves 5 traits,</p> <pre><code>!DEFINE Time Trait 1 3 5 10 20</code></pre> |
| <code>!DDF [i]</code>   | <p>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</i>: <i>i</i> = -1 suppresses computation, <i>i</i> = 1 and <i>i</i> = 2 compute the denominator d.f. using numerical and algebraic methods respectively. If <code>!DDF i</code> is omitted, <i>i</i> = -1 is assumed except for small jobs (&lt; 10 parameters, &lt; 500 fixed effects, &lt; 10,000 equations and &lt; 1 Gbyte workspace) when <i>i</i> = 2.</p> <p>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 <code>!MAXIT</code> is 1 or for multivariate analysis.</p>  |
| <code>!FCON</code>  | <p>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 <code>.aov</code> 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 <code>A.C</code> cannot be adjusted for <code>A.B.C</code>) or implicitly (e.g. term <code>REGION</code> cannot be adjusted for</p>   |

## 5.8 Job control qualifiers

| qualifier                                 | action   |
|---|--|
|   | <p>LOCATION when locations are actually nested in regions although they are coded independently). !FOWN in Table 5.8 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.</p>  |
| !MAXIT <i>n</i>                           | <p>sets the maximum number of iterations; the default is 10 for traditional models, more for general models. <b>ASReml</b> iterates for <i>n</i> iterations unless convergence is achieved first. Convergence is presumed when the <b>REML</b> log-likelihood changes less than <math>0.002 * \text{current iteration}</math> number and the individual variance parameter estimates change less than 1%.</p> <p>If the job has not converged in <i>n</i> iterations, use the !CONTINUE qualifier to resume iterating from the current point.</p> <p>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, <b>ASReml</b> 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 <b>ASReml</b> after one more iteration (during which predictions will be formed).</p> <p>On case sensitive operating systems (e.g. 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).</p> <p>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).</p> <p>If !MAXIT 1 is used and an Unstructured Variance model is fitted, <b>ASReml</b> 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, <b>ASReml</b> tests the adequacy of the reduced parameterization.</p> |
| !SUM                                      | <p>causes <b>ASReml</b> 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.</p>   |
| !X <i>v</i> !Y <i>v</i> !G <i>v</i> !JOIN | <p>is used to plot the (transformed) data. Use !X to specify the <i>x</i> variable,</p> <p>!Y to specify the <i>y</i> variable and !G to specify a grouping variable. !JOIN joins the points when the <i>x</i> value increases between consecutive records. The grouping variable may be omitted for a simple scatter plot. Omit</p> <p>!Y <i>y</i> to produce a histogram of the <i>x</i> variable.</p> <p>For example</p> <pre>!X age !Y height !G sex</pre> <p>Note that the graphs are only produced in the graphics versions of <b>ASReml</b> (Section 11.3.4).</p> <p>For multivariate repeated measures data, <b>ASReml</b> can plot the response profiles if the first response is nominated with the !Y qualifier and the following analysis is of the multivariate data. <b>ASReml</b> assumes the response variables are in contiguous fields and are equally spaced. For example,</p>  |

## 5.8 Job control qualifiers

| qualifier | action  |
|-----------|---|
|           | <p>Response profiles</p> <p>Treatment !A Y1 Y2 Y3 Y4 Y5</p> <p>rat.asd !Y Y1 !G Treatment !JOIN</p> <p>Y1 Y2 Y3 Y4 Y5 ~ Trait Treatment Trait.Treatment</p> |

Table 5.7: List of occasionally used job control qualifiers

| qualifier            | action  |
|----------------------|---|
| !ASMV <i>n</i>       | <p>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 <a href="#">Chapter 8</a> for a complete discussion.</p> <p>The data is presumed arranged in lots of <i>n</i> records where <i>n</i> is the number of <i>traits</i>. 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.</p> |
| !ASUV                | <p>allows you to have an error variance other than <math>I \otimes \Sigma</math> where <math>\Sigma</math> is the unstructured (US, see <a href="#">Table 7.10</a>) variance structure, if the data is presented in a multivariate form. If there are <i>missing values</i> 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 <a href="#">Chapter 8</a> for more information.</p> <p>This option is used for repeated measures analysis when the variance structure required is not the standard multivariate unstructured matrix.</p>   |
| !DESIGN              | <p>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 <a href="#">Section 7.8.2</a>.</p>  |
| !DISPLAY <i>n</i>    | <p>is used to select particular graphic displays. In spatial analysis of field trials, four graphic displays are possible (see <a href="#">Section 14.4</a>). Coding these</p> <p>1=variogram</p> <p>2=histogram</p> <p>4=row and column trends</p> <p>8=perspective plot of residuals,</p> <p>set <i>n</i> to the sum of the codes for the desired graphics. The default is 9=1+8.</p> <p>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 (<a href="#">Section 11.3.4</a> on graphics) for how to save the graphs in a file for printing.</p> <p>Use !NODISPLAY to suppress graphic displays.</p>   |
| !EPS                 | <p>sets hardcopy graphics file type to .eps.</p>  |
| !G <i>v</i>          | <p>is used to set a grouping variable for plotting, see !X.</p>   |
| !GKRIGE [ <i>p</i> ] | <p>controls the expansion of !PVAL lists for fac(<i>X</i>, <i>Y</i>) model terms. For kriging</p>   |

## 5.8 Job control qualifiers

| qualifier  | action  |
|--|---|
|  | <p>prediction in 2 dimensions (<math>X, Y</math>), 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 <math>X</math> and <math>Y</math> using two <code>!PVAL</code> statements. Normally, predict points will be defined for all combinations of <math>X</math> and <math>Y</math> 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.</p> <p>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, <b>ASReml</b> will ignore the extra PVAL points unless either <code>!FINAL</code> or <code>!GKRIGE</code> are set, to save processing time.</p>   |
| <code>!GROUPFACTOR <math>t</math> <math>v</math> <math>p</math></code>   | <p>the <code>!GROUPFACTOR</code> qualifier, like <code>!SUBSET</code>, must appear on a line by itself after the data line and before the model line. Its purpose is to define a factor <math>t</math> by merging levels of an existing factor <math>v</math>. The syntax is <code>!GROUPFACTOR &lt;Group_factor&gt; &lt;Exist_factor&gt; &lt;new codes&gt;</code></p> <p>for example</p> <pre>!GROUPFACTOR Year YearLoc 1 1 1 2 2 3 3 3 4 4</pre> <p>forms a new factor <code>Year</code> with 4 levels from the existing factor <code>YearLoc</code> with 10 levels.</p> <p>Alternatively, <code>Year</code> could be formed by data transformation</p> <pre>Year * !=YearLoc !set 1 1 1 2 2 3 3 3 4 4 !L 2001 2002 2003 2004</pre>   |
| <code>!IDLIMIT <math>v</math></code>   | <p>If an AR1 structure is specified in a residual statement like</p> <pre>residual sat(Block).ar1(row).ar1v(col)</pre> <p>some blocks may have insufficient rows or columns to meaningfully estimate the autoregression parameters. This qualifier fixes the parameter at its initial value (0.1 by default) if there are no more than <math>v</math> levels in the structure. If the qualifier is unspecified, <math>v</math> is 1. If no argument is specified the default is 4.</p>  |
| <code>!JOIN</code>   | <p>is used to join lines in plots, see <code>!X</code>.</p>   |
| <code>!MBF mbf(<math>v, n</math>) <math>f</math></code><br><code>[!FACTOR ]</code><br><code>[!FIELD <math>s</math>]</code><br><code>[!KEY <math>k</math>]</code><br><code>[!NOKEY ]</code><br><code>[!RENAME <math>t</math>]</code><br><code>[!RFIELD <math>r</math>]</code><br><code>[!SKIP <math>k</math>]</code><br><code>[!SPARSE ]</code> | <p>specified on a separate line after the datafile line predefines the model term <code>mbf(<math>v, n</math>)</code> as a set of <math>n</math> covariates indexed by the data values in variable <math>v</math>. MBF stands for My Basis Function and uses the same mechanism as the <code>leg()</code>, <code>pol()</code> and <code>spl()</code> 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 <math>f</math> should contain <math>1+n</math> fields where the first field, the <i>key</i> field, contains the values which are in the data variable or at which prediction is required, and the remaining <math>n</math> fields define the corresponding covariate values. If <math>n</math> is omitted, all fields after the <i>key</i> field, are taken unless <code>!FACTOR</code> is specified for which <math>n</math> is 1 and the covariate values are treated as coding for a multilevel factor. Set <math>n</math> to 1 to read just one field from the data file. Also note that the file may be a binary file (e.g. formed in a previous run using <code>!SAVE</code>).</p> <p><code>!RENAME <math>t</math></code> changes the name of the term from <code>mbf(...)</code> to the new name <math>t</math>. This is necessary when several <code>mbf(...)</code> terms are being defined which would otherwise have the same name/label. For example</p> <pre>!MBF mbf(entry) mlib/m35.csv !RENAME Marker35</pre> |

## 5.8 Job control qualifiers

| qualifier                     | action  |
|-------------------------------|---|
|                               | <p>If the <i>key</i> values are the ordered sequence <math>1 : N</math>, the <i>key</i> field may be omitted if <code>!NOKEY</code> is specified. If the <i>key</i> is not in the first field, its location can be specified with <code>!KEY <i>k</i></code>. If extracting a single covariate from a large set of covariates in the file, the specific field to extract can be given by <code>!FIELD <i>s</i></code> in absolute terms, or relative to the <i>key</i> field by <code>!RFIELD <i>r</i></code>. For example</p> <pre>!MBF mbf(variety,1) markers.csv !key 1 !RFIELD 35 !RENAME M35</pre> <p><code>!SKIP <i>k</i></code> requests the first <i>k</i> lines of the file be ignored.</p> <p><code>!SPARSE</code> can be used when the covariates are predominately zero. Each <i>key</i> value is followed by as many <i>column,value</i> pairs as required to specify the non-zero elements of the design for that value of <i>key</i>. The pairs should be arranged in increasing order of <i>column</i> within rows. The rows may be continued on subsequent lines of the file provided incomplete lines end with a COMMA.</p> <p>This file may now be a binary format file, with file extension <code>.bin</code> indicating 32bit real binary numbers and <code>.dbl</code> indicating 64bit real binary values. Files with these formats can be easily created in a preliminary run using the <code>!SAVE</code> 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</p> <pre>!CYCLE 1:1000 !MBF mbf(Geno) markers.dbl !key 1 !RFIELD \$I !RENAME M\$I ...      !r M\$I</pre> <p><b>Restrictions</b> The <i>key</i> field MUST be numeric. In particular, if the data field it relates to is either an <code>!A</code> or <code>!I</code> encoded factor, the original (uncoded) level labels may not be 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 <b>ASReml</b> when the MBF file is processed, although the user can/must anticipate what it will be.</p> <p><b>Comment</b> If this MBF process is to be used repeatedly, for example to process a large set of marker variables in conjunction with <code>!CYCLE</code>, processing will be much faster if the markers variables are in separate files. <b>ASReml</b> 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 <code>!SAVE</code>.</p> |
| <code>!MP <i>p</i></code>     | specifies <i>p</i> the maximum number of threads to be used with OMP parallel processing. <b>ASReml</b> will use all threads available up to the maximum <i>p</i> (default 16). <b>ASReml</b> reports in the <code>.asr</code> file the number of used and available threads.   |
| <code>!MVINCLUDE</code>       | when missing values occur in the design <b>ASReml</b> will report this fact and abort the job unless <code>!MVINCLUDE</code> is specified (see Section 6.9); then missing values are treated as zeros. Use the <code>!DV</code> transformation to drop the records with the missing values.   |
| <code>!MVREMOVE</code>        | instructs <b>ASReml</b> to discard records which have missing values in the design matrix (see Section 6.9).  |
| <code>!NODISPLAY</code>       | 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 <code>N</code> (see Section 11.3.4 on graphics). The text version of the graphics is still written to the <code>.res</code> file.  |
| <code>!PVAL <i>v p</i></code> | is a mechanism for specifying the particular points to be predicted for covariates modelled using <code>fac(<i>v</i>)</code> , <code>leg(<i>v,k</i>)</code> , <code>spl(<i>v,k</i>)</code> and <code>pol(<i>v,k</i>)</code> . The points are specified here so that they can be included in the appropriate design matrices. <i>v</i> is the name of a data field. <i>p</i> is the list of values at which prediction is required. See  |

## 5.8 Job control qualifiers

| qualifier                 | action   |
|---------------------------|--|
|                           | !GKRIGE for special conditions pertaining to $\text{fac}(x, y)$ prediction.  |
| !PVAL <i>f vlist</i>      | is used to read <i>predict_points</i> for several variables from a file <i>f</i> . <i>vlist</i> is the names of the variables having values defined. If the file contains unwanted fields, put the pseudo variate label <i>skip</i> in the appropriate position in <i>vlist</i> to ignore them. The file should only have numeric values. <i>predict_points</i> cannot be specified for design factors.  |
| !SECTION <i>v</i>         | <p>specifies the variable in the data that defines the data sections. This qualifier enables <b>ASReml</b> to check that sections have been correctly dimensioned. Further, when the model term <i>mv</i> is included in the model and !ROWFACTOR and !COLUMNFACTOR are defined, <b>ASReml</b> 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)</p> <pre> Basic MET analysis filling out row and column grid seq          # sequential id col *        # columns coded 1... row *        # rows coded 1... chks 7       # check names test 336 variety !A yld !*.01 site 3       # sites coded 1,2,3 met.dat !SKIP 1 !SECTION site !ROWFACTOR row !COLFACTOR col yld ~ site !r variety site.variety !f mv residual sat(site).ar1(row).ar1(col) </pre>   |
| !SPLINE <i>spl(v,n) p</i> | <p>defines a spline model term with an explicit set of knot points. The basic form of the spline model term, <i>spl(v)</i>, is defined in Table 6.1 where <i>v</i> is the underlying variate. The basic form uses the unique data values as the knot points. The extended form is <i>spl(v,n)</i> which uses <i>n</i> knot points. Use this !SPLINE qualifier to supply an explicit set of <i>n</i> knot points (<i>p</i>) for the model term <i>t</i>. Using the extended form without using this qualifier results in <i>n</i> 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.</p> <p>When knot points are explicitly supplied, they should be in increasing order and adequately cover the range of the data or <b>ASReml</b> 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 <b>ASReml</b> will 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.</p> |
| !STEP <i>r</i>            | 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 <i>t v p</i>    | this qualifier must appear on a line by itself after the data line and before the model line. This qualifier forms a new group factor ( <i>t</i> ) derived from an existing group factor ( <i>v</i> ) by selecting a subset ( <i>p</i> ) of its variables. A subgroup factor may not be used in a PREDICT or TABULATE directive.   |
| !SUBSET <i>t v p</i>      | this qualifier forms a new factor ( <i>t</i> ) derived from an existing factor ( <i>v</i> ) by selecting a subset ( <i>p</i> ) 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   |



## 5.8 Job control qualifiers

| qualifier                  | action  |
|----------------------------|---|
|                            | <p>datafile line but before the linear model. <i>e.g.</i></p> <pre>!SUBSET EnvC Env 3 5 8 9 :15 21 33</pre> <p>defines a reduced form of the factor <code>Env</code> just selecting the environments listed. It might then be used in the model in an interaction. A subset factor can be used in a <code>TABULATE</code> directive but not in a <code>PREDICT</code> directive.</p> <p>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 <code>Trait</code> in a multivariate analysis provided it correctly identifies the number of levels of <code>Trait</code> either by including the last trait number, or appending sufficient zeros. Thus, if the analysis involves 5 traits,</p> <pre>!SUBSET Trewe Trait 1 3 4 0 0</pre>  |
| <code>!TAU <i>f</i></code> | <p>specifies that ASReml saves any regression coefficient associated with covariate <i>f</i> as a string named <code>Tau</code> so that the value can be used to adjust data in the next run. For example, the following code performs a two-stage analysis properly adjusting for plant height. First, we specify <code>!TAU x</code> to nominate the covariate whose coefficient is to be captured, or the position number of the covariate. The default position is 1. Second, we specify <code>\$Tau</code> to substitute the coefficient into the job code.</p> <pre>!RENAME 1 !ARG 1 2 Slate Hall 1976 Cereal trial   rep      6   latrow   30   latcol   30   variety  25   yield   fldrow   15   fldcol   10   PlantHt   adjyield !=PlantHt !*\$Tau !*-1 !+yield shftau.asd !SKIP 1 !NOD !TAU PlantHt !DOPART \$1 !PART 1 #Fitting AR1.AR1 - Gamma Scale adjyield ~ mu PlantHt mv !r var fldr fldc res ar1(fldr).ar1v(fldc) !PART 2 #Fitting AR1.AR1 adjyield ~ mu var mv !r fldr fldc res ar1(fldr).ar1v(fldc) PREDICT var !TWOSTAGEWTS</pre> <p>This job fits two models. In the first, <code>\$Tau</code> has a value of 1 when used to calculate <code>adjyld</code> but <code>adjyld</code> is not used in the first model. The coefficient value is updated at the end of the first model fit to the regression coefficient obtained in that fit for plantheight. In the second model, we analyse <code>adjyld</code> which has been adjusted for plantheight and the residual degrees of freedom reduced by 1 to take account of the plantheight adjustment. This example does not take into account the mean plantheight.</p> |
| <code>!WMF</code>          | <p>sets hardcopy graphics file type to <code>.wmf</code>.</p>   |



Table 5.8: List of rarely used job control qualifiers

| qualifier                     | action   |
|-------------------------------|--|
| <code>!ARLIMIT [p]</code>     | where the correlation parameter in an <code>ar1()</code> model goes to the boundary ( $\pm 0.999$ ); the qualifier resets the boundary limit for the magnitude of the AR parameter to $p$ (default 0.75).  |
| <code>!AILOADINGS i</code>    | <p>controls modification to AI updates of loadings in extended Factor Analytic models. After <b>ASReml</b> calculates updates for variance parameters, it checks whether the updates are reasonable and sometimes reduces them over and above any <code>!STEPSIZE</code> 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.</p> <p>Unless the user gives constraints, <b>ASReml</b> sets them and rotates the loadings each iteration. When <code>!AILOADINGS i</code> is specified, it also prevents AI updates of some loadings during the first <math>i</math> iterations. For <math>f(&gt;1)</math> factors, only the last factor is estimated (conditional on the earlier ones) in the first <math>f-1</math> iterations. Then pairs including the last are estimated until iteration <math>i</math>.</p> <p>If <code>!AILOADINGS</code> is not specified and <code>!CONTINUE</code> is used and initializes the XFA model from a lower order, the <math>i</math> parameter is set internally.</p>   |
| <code>!AIPENALTY [p]</code>   | <p>The algorithm for updating loadings in factor analytic models has been improved. This builds on an earlier change implemented in <b>ASReml 4</b> which modified updates to loadings, but proved too conservative causing such jobs to take too many iterations to converge. The motivation for change was that the original update procedure sometimes produced unreasonable updates, or otherwise came near to convergence and then drifted away. The present procedure is to modify the average information matrix by increasing the diagonal elements pertaining to loadings by a percentage, <math>p</math>. The default is to start with <math>p = 10\%</math> and reduce it by 1 or 2% each iteration down to 1%. If the starting values are poor, 10% may not be a sufficient initial retardation. If it appears the updates are unreasonable, <b>ASReml</b> will increase the value of <math>p</math> by 10% and then continue. The user can set the initial value of <math>p</math> with the qualifier <code>!AIPENALTY p</code>. After the penalty has reduced to 1%, it is further reduced to 0.2%. The qualifier can be used to set <math>p</math> to 0 if desired. The value of <math>p</math> can be monitored by using the <code>!LOGFILE</code> and <code>!DEBUG</code> command line qualifiers and searching the <code>.asl</code> file for the string <code>XFAIF</code> (XFA Inflation Factor).</p>  |
| <code>!AISINGULARITIES</code> | <p>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. <b>ASReml</b> now aborts processing if such singularities appear unless the <code>!AISINGULARITIES</code> qualifier is set. Which particular parameter is singular is reported in the variance component table printed in the <code>.asr</code> file.</p> <p>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.</p> <p>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.</p> <p>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 <code>!AISINGULARITIES</code> qualifier.</p> |

## 5.8 Job control qualifiers

| qualifier           | action   |
|---------------------|--|
| !BMP                | sets hardcopy graphics file type to .bmp.  |
| !BRIEF [ <i>n</i> ] | 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 <i>n</i>      | <p>is used to report fitted values from the model assuming the initial values of the variance, to the .sln file and then stop, 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</p> <ul style="list-style-type: none"> <li><math>n = 1</math>: forming the estimates of the vector of fixed and random effects by matrix inversion</li> <li><math>n = 2</math>: forming the estimates of the vector of fixed and random effects, REML log-likelihood and residuals (this is the default)</li> <li><math>n = 3</math>: forming the estimates of the vector of fixed and random effects, REML log-likelihood, residuals and inverse coefficient matrix.</li> </ul> <p>For arguments 4 and 10:19, ASReml forms the mixed model equations and solves them iteratively to obtain solutions for the fixed and random effects. The options are</p> <ul style="list-style-type: none"> <li><math>n = 4</math>: forming the estimates of the vector of fixed and random effects using the Preconditioned Conjugate Gradient (PCG) Method (Mrode, 2005)</li> <li><math>n = 10:19</math> forming the estimates of the vector of fixed and random effects by Gauss-Seidel iteration of the mixed model equations, with relaxation factor <math>n/10</math>.</li> </ul> <p>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 <math>1e^{-10}</math>. Gauss-Seidel iteration is generally much slower than the PCG method.</p> <p>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 <math>n=3</math>. Residuals are not available if <math>n = 1</math>. Use of <math>n = 3</math> or <math>n = 2</math> will halve the processing time when compared to the alternative of using !MAXIT 1 rather than a !BLUP <i>n</i> qualifier. However, !MAXIT 1 does result in complete and correct output.</p> |
| !DENSE <i>n</i>     | 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 <i>n</i> 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. <i>n</i> should be kept small (<100) for faster processing.   |
| !DF <i>n</i>        | alters the error degrees of freedom from $v$ to $v+n$ . This qualifier might be used when analysing pre-adjusted data to reduce the degrees of freedom ( <i>n</i> negative) or when weights are used in lieu of actual data records to supply error information ( <i>n</i> positive). The degrees of freedom are only used in the calculation of the residual variance when the averaged data is fully fitted in a univariate single site analysis. The option will have no effect in analyses with multiple error variances (for sites or traits) other than  |

## 5.8 Job control qualifiers

| qualifier  | action  |
|--|---|
|  | <p>in the reported degrees of freedom. Use <code>!ADJUST <i>r</i></code> rather than <code>!DF <i>n</i></code> if <i>r</i> is not a whole number. Use with <code>!YSS <i>r</i></code> to supply extra variance when summarized data is analysed with weights and variance has been lost as a result.</p>  |
| <code>!EM <i>n</i></code><br><code>!PXEM <i>n</i></code> | <p>requests <b>ASReml</b> use Expectation-Maximization (EM) rather than Average Information (AI) updates when the AI updates would make a <b>US</b> structure non-positive definite. This only applies to <b>US</b> structures. When <code>!GP</code> is associated with a <b>US</b> structure, <b>ASReml</b> 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 an EM update each iteration even though <code>!EM</code> is omitted.</p> <p>EM is notoriously slow at finding the solution and <b>ASReml</b> includes several modified schemes, discussed by Cullis <i>et al.</i> (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.</p> <p>The options are</p> <p>With odd arguments <code>!EM</code> initiates sequences of EM updates</p> <ul style="list-style-type: none"> <li>• <code>!EM 1</code> Standard EM plus 10 local EM steps</li> <li>• <code>!EM 3</code> Standard EM plus 10 local EM steps*</li> <li>• <code>!EM 5</code> Standard EM only</li> <li>• <code>!EM 7</code> Standard EM plus 1 local EM step</li> </ul> <p>With even arguments <code>!PXEM</code> initiates sequences of EM and PXEM updates</p> <ul style="list-style-type: none"> <li>• <code>!PXEM 2</code> Standard EM plus 10 local PXEM steps</li> <li>• <code>!PXEM 4</code> Standard EM plus 10 local PXEM steps*</li> <li>• <code>!PXEM 6</code> Single local PXEM</li> <li>• <code>!PXEM 8</code> Standard EM plus 1 local PXEM steps</li> </ul> <p>* Options 3 and 4 cause all <b>US</b> structures to be updated by (PX)EM if any particular one requires EM updates.</p> <p>The test of whether the AI updated matrix is positive definite 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 <math>10^{-7}</math> (this value is consistent with the multiple partial correlation of the first variable with the rest being greater than 0.9999999, <b>ASReml</b> fixes the matrix at that point and estimates any other parameters conditional on these values. To proceed 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.</p> |
| <code>!EQORDER <i>o</i></code>                           | <p>modifies the algorithm used for choosing the order for solving the mixed model equations. A new algorithm devised for <b>Release 2</b> is now the default and is formally selected by <code>!EQORDER 3</code>. The algorithm used for <b>Release 1</b> is essentially that selected by <code>!EQORDER 1</code>. The new order is generally superior. <code>!EQORDER -1</code> instructs <b>ASReml</b> 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 by a model containing terms associated with a generalized relationship matrix and an additive relationship matrix</p>  |

## 5.8 Job control qualifiers

| qualifier             | action   |
|-----------------------|--|
|                       | <code>y ~ mu !r str(grm(ind) ind)</code>   |
|                       | <code>grm(ind)</code> invokes a dense sparse matrix and <code>grm(ind)</code> has a sparse structured inverse of an additive relationship matrix in the mixed model equations, and <code>str()</code> ensures that the model terms are not reordered when the equations are constructed. While <code>!EQORDER 3</code> generates a more sparse solution, <code>!EQORDER -1</code> runs faster.   |
| <code>!EXTRA n</code> | forces another <code>mod(n, 10)</code> rounds of iteration after apparent convergence. The default for <code>n</code> is 1. This qualifier has lower priority than <code>!MAXIT</code> and <code>ABORTASR.NOW</code> (see <code>!MAXIT</code> for details).<br><br>Convergence is judged by changes in the REML log-likelihood value and variance parameters. However, sometimes the variance parameter convergence criteria have not been satisfied.  |
| <code>!FOWN</code>    | allows the user to specify the test reported in the F-con column of the Wald F Statistics table. It has the form<br><br><code>!FOWN terms to test ; background terms</code><br><br>placed on a separate line immediately after the model line. Multiple <code>!FOWN</code> statements should appear together. It generates a Wald F statistic for each model term in <i>terms to test</i> which tests its contribution after all other terms in <i>terms to test</i> and <i>background terms</i> , 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.<br><br>For example<br><br><code>!FOWN A B C ; mu</code><br><br>would request the Wald F statistics based on (see Section 2.5.2)<br><br><code>R(A   mu B C sparse)</code><br><code>R(B   mu A C sparse)</code><br><code>R(C   mu A B sparse)</code><br><br>And<br><br><code>!FOWN A.B B.C A.C ; mu A B C</code><br><br>would request the Wald F statistics<br><br><code>R(A.B   mu A B C B.C A.C sparse)</code><br><code>R(B.C   mu A B C A.B A.C sparse)</code><br><code>R(A.C   mu A B C A.B B.C sparse)</code><br><br>And<br><br><code>!FOWN A.B.C ; mu A B C A.B B.C A.C</code><br><br>would request the Wald F statistic<br><br><code>R(A.B.C   mu A B C A.B A.C B.C sparse)</code> |

### 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 have more or less NumDF than are present in the full rank parameterization because aliased effects are reordered, it cannot 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

## 5.8 Job control qualifiers

| qualifier          | action  |
|--------------------|---|
|                    | <p>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.</p> <ul style="list-style-type: none"> <li>Any model terms in the !FOWN lists which do not appear in the actual model, are ignored without flagging an error.</li> <li>Any model terms which are omitted from !FOWN statements are tested with the usual conditional test.</li> <li>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, . . .</li> </ul> <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.</p>  |
| !FREEGH <i>p</i>   | sets number of iterations for a parameter to be held using the !HOLD qualifier (see Section 7.7.4).   |
| !GDENSE            | is used to have the first random term included in the <i>dense</i> equations if it is a GRM/GIV variance structure. This will result in faster processing when the GRM (inverse) matrix is not sparse.  |
| !GINDEX <i>t p</i> | <p>This qualifier !GINDEX <i>ModelTerm Coefficients</i> should be placed after the datafile line and before the model line. The qualifier is motivated by wishing to predict effects indices that are linear combinations of objective traits that are not measured. In principle these objective traits could be included directly in a mixed model but it is sometimes more computationally feasible to calculate the predicted values by using the idea of a selection index and form the index from the predictions of the measured trait. If the <i>i</i>-th index for an individual is <math>\alpha_i \mathbf{u}_0</math> with linear combination <math>\alpha_i</math> of objective traits effects <math>\mathbf{u}_0</math> then this index can be predicted with <math>\mathbf{b}_i \mathbf{u}_m</math> with <math>\mathbf{b}_i = \alpha_i \mathbf{G}_{om} \mathbf{G}_{mm}^{-1}</math> where <math>\mathbf{G}_{om}</math> is the covariance between objective and measured trait effects and <math>\mathbf{G}_{mm}</math> is the variance of measured trait effects and <math>\mathbf{u}_0</math> is the prediction of the effects for the measured traits. Note that this qualifier is envisaged to be used to provide predictions of effects using one round of iteration and using known values of the variance parameters. For example,</p> <pre> harvey.ped !SKIP 1 !ALPHA harvey.dat !SKIP 1 !MAXIT 1 !GINDEX us(Trait).nrm(animal), 0.25 0.75, # Index 1 0.70 0.30 # Index 2 !ASSIGN INITS 19.96 8.985 117.2 !ASSIGN INITR 134.8 -66.45 655.2 Y1 Y3 ~ Trait Trait.line !r us(Trait \$INITS).nrm(animal) residual id(units).us(Trait \$INITR) </pre> <p>generates predicted values for 2 indices using animal effect predicted values for traits ADG and Y3. The term <code>us(Trait)</code> identifies <math>\mathbf{G}_{mm}</math>. Forty coefficients can be supplied and the number is a multiple of the number of traits. The computed selection and prediction error standard errors are written to the .sli file. Note that the prediction error standard errors are calculated from the prediction of <math>\mathbf{b}_i \mathbf{u}_m</math>. The prediction error of <math>\mathbf{b}_i \mathbf{u}_m</math> also includes an extra term dependent on <math>(\mathbf{b}_i \mathbf{u}_m - \alpha_i \mathbf{u}_0)</math>, the extra variation from predicting the indices using measured traits instead of objective traits. This variation depends on <math>\mathbf{G}_{oo} - \mathbf{G}_{om} \mathbf{G}_{mm}^{-1} \mathbf{G}'_{mo}</math> but at present this is not calculated by ASReml.</p> |
| !GLMM [ <i>n</i> ] | sets the number of inner iterations performed when an 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   |

## 5.8 Job control qualifiers

| qualifier   | action   |
|---|--|
|   | first round and 2 in subsequent rounds of the outer iteration. Set $n$ to 2 or more to increase the number of inner iterations.  |
| <code>!HPGL [2]</code>  | sets hardcopy graphics file type to HP GL. An argument of 2 sets the hardcopy graphics file type to HP GL 2.   |
| <code>!HOLD [list]</code>   | allows the user to temporarily fix the parameters listed. Each variance structure parameter is allocated a number internally. These numbers are reported in the <code>.tsv</code> file and some are reported in the structure input section of the <code>.asr</code> file. The list should be in increasing order using colon to indicate a sequence, step size is 1. Use <code>!FREEGH</code> to control the number of iterations <code>!HOLD</code> applies to. This qualifier is intended for use when some variance parameters have ‘good’ starting values and other not.<br><br>For example<br><code>!HOLD 1:20 30:40</code>  |
| <code>!LAST &lt;factor<sub>1</sub>&gt;</code><br><code>&lt;lev<sub>1</sub>&gt; [&lt;fac<sub>2</sub>&gt;</code><br><code>&lt;lev<sub>2</sub>&gt; &lt;fac<sub>3</sub>&gt;</code><br><code>&lt;lev<sub>3</sub>&gt;]</code> | limits the order in which equations are solved in <b>ASReml</b> by forcing equations in the sparse partition involving the first <code>&lt;lev<sub>i</sub>&gt;</code> equations of <code>&lt;factor<sub>i</sub>&gt;</code> 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 <b>ASReml</b> will be consistent in which effects are identified as singular. The test example had<br><br><code>!r Anim Litter !f HYS</code><br><br>where genetic groups were included in the definition of <code>Anim</code> .<br><br>Consequently, there were 5 singularities in <code>Anim</code> . The default reordering allows those singularities to appear anywhere in the <code>Anim</code> and <code>HYS</code> terms. Since 29 genetic groups were defined in <code>Anim</code> , <code>!LAST Anim 29</code> forces the genetic group equations to be absorbed last (and therefore incorporate any singularities). In the more general model fitting<br><br><code>!r Tr.Anim Tr.Lit !f Tr.HYS</code><br><br>without <code>!LAST</code> , the location of singularities will almost surely change if the G structures for <code>Tr.Anim</code> or <code>Tr.Lit</code> are changed, invalidating Likelihood Ratio tests between the models. |
| <code>!NOZEROVC</code>  | suppresses the default action of fixing simple variance components exactly at 0.0 rather than fixing them at a small value. The default action applies to simple components, such as elements in a <code>diag()</code> structure and $\Psi$ values in the extended factor analytic models.   |
| <code>!OUTLIER</code>   | performs the outlier check described in Section 2.4.2. This can have a large time penalty in large models.   |
| <code>!OWN <math>f</math></code>  | supplies the name of a program supplied by the user in association with the <code>OWN</code> variance model (Section 7.7.3).   |
| <code>!PRINT <math>n</math></code>  | causes <b>ASReml</b> to print the transformed data file to <code>basename.asp</code> . If <ul style="list-style-type: none"> <li><math>n &lt; 0</math>, data fields <math>1 \dots \text{mod}(n)</math> are written to the file</li> <li><math>n = 0</math>, nothing is written</li> <li><math>n = 1</math>, all data fields are written to the file if it does not exist</li> <li><math>n = 2</math>, all data fields are written to the file overwriting any previous contents</li> <li><math>n &gt; 2</math>, data fields <math>n \dots t</math> are written to the file where <math>t</math> is the last defined column.</li> </ul>   |
| <code>!PNG</code>   | sets hardcopy graphics file type to <code>.png</code> .  |
| <code>!PS</code>  | sets hardcopy graphics file type to <code>.ps</code> .   |

## 5.8 Job control qualifiers

| qualifier                                       | action  |
|---|---|
| <code>!PVSFORM <i>n</i></code>                  | <p>modifies the format of the tables in the <code>.pvs</code> file and changes the file extension of the file to reflect the format. See <code>!TXTFORM</code> for more detail.</p> <ul style="list-style-type: none"> <li>• <code>!PVSFORM 1</code> is TAB separated: <code>.pvs</code>→<code>_pvs.txt</code></li> <li>• <code>!PVSFORM 2</code> is COMMA separated: <code>.pvs</code>→<code>_pvs.csv</code></li> <li>• <code>!PVSFORM 3</code> is Ampersand separated: <code>.pvs</code>→<code>_pvs.tex</code></li> </ul>   |
| <code>!RESIDUALS [2]</code>                     | <p>instructs <b>ASReml</b> to write the transformed data and the residuals to a binary file. The residual is the last field. The file <code>basename.srs</code> is written in single precision unless the argument is 2 in which case <code>basename.drs</code> is written in double precision. Factor names are held in a <code>.vll</code> file: see <code>!SAVE</code> below. If a Generalized Linear Model was fitted, the weights are also written to this file after the residuals.</p> <p>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 <code>units</code> part added to it when <code>units</code> is also fitted. The <code>.drs</code> file could be renamed (with extension <code>.dbl</code>) and used for input in a subsequent run.</p>  |
| <code>!S2LIMIT</code>                           | <p>limits the size of the residual variance in a multi-environment spatial analysis to a tenth of the average spatial residual variance across all environments, fixing it at that small but not unreasonable value if the updated value is smaller. The motivation is that occasionally in unreplicated trials with a nugget variance also fitted, the spatial variance is poorly estimated and will otherwise go to almost zero which may difficult the estimation of other terms in the model.</p>   |
| <code>!SAVE <i>n</i></code>                     | <p>instructs <b>ASReml</b> to write the data to a binary file. The file <code>asrdata.bin</code> is written in single precision if the argument <i>n</i> is 1 or 3; <code>asrdata.dbl</code> is written in double precision if the argument <i>n</i> 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).</p> <p>When either <code>!SAVE</code> or <code>!RESIDUALS</code> is specified, <b>ASReml</b> saves the factor level labels to a <code>basename.vll</code> and attempts to read them back when data input is from a binary file. Note that if the job <code>basename</code> changes between runs, the <code>.vll</code> file will need to be copied to the new <code>basename</code>. If the <code>.vll</code> file does not match the factor structure (<i>i.e.</i> the same factors in the same order), reading the <code>.vll</code> file is aborted.</p>   |
| <code>!SCREEN [<i>n</i>] [!SMX <i>m</i>]</code> | <p>performs a 'Regression Screen', a form of all subsets regression. For <i>d</i> model terms in the DENSE equations, there are <math>2^d - 1</math> possible submodels. Since for <math>d &gt; 8</math>, <math>2^d - 1</math> is large, the submodels explored are reduced by the parameters <i>n</i> and <i>m</i> so that only models with at least <i>n</i> (default 1) terms but no more than <i>m</i> (default 6) terms are considered. The output is a report to the <code>.asr</code> file with a line for every submodel showing the sums of squares, degrees of freedom and terms in the model (see Section 14.3.1). There is a limit of <math>d = 20</math> model terms in the screen. <b>ASReml</b> 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 <code>!SCREEN 3 !SMX 3</code>. The number of models evaluated quickly increases with <i>d</i> but <b>ASReml</b> has an arbitrary limit of 900 submodels evaluated. Use the <code>!DENSE</code> qualifier to control which terms are screened. The screen is conditional on all other terms (those in the SPARSE equations) being present.</p> |
| <code>!SLNFORM [<i>n</i>]</code>                | <p>modifies the format of the <code>.sln</code> file</p> <ul style="list-style-type: none"> <li>• <code>!SLNFORM -1</code> prevents the <code>.sln</code> file from being written.</li> <li>• <code>!SLNFORM 1</code> is TAB separated: <code>.sln</code> becomes <code>_sln.txt</code></li> <li>• <code>!SLNFORM 2</code> is COMMA separated: <code>.sln</code> becomes <code>_sln.csv</code></li> </ul>   |

## 5.8 Job control qualifiers

| qualifier                   | action  |
|-----------------------------|---|
|                             | <ul style="list-style-type: none"> <li>• <code>!SLNFORM 3</code> is Ampersand separated: <code>.sln</code> becomes <code>_sln.tex</code></li> </ul> <p>Note that, extra significant digits are reported when <code>!SLNFORM</code> is set, and expanded labelling of the levels in interactions is used because field width is no longer restricted. See <code>!TXTFORM</code> for more detail.</p>   |
| <code>!SPATIAL</code>       | 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 <code>.res</code> file.   |
| <code>!TABFORM [n]</code>   | <p>controls form of the <code>.tab</code> file</p> <ul style="list-style-type: none"> <li>• <code>!TABFORM 1</code> is TAB separated: <code>.tab</code> becomes <code>_tab.txt!</code></li> <li>• <code>!TABFORM 2</code> is COMMA separated: <code>.tab</code> becomes <code>_tab.csv!</code></li> <li>• <code>!TABFORM 3</code> is Ampersand separated: <code>.tab</code> becomes <code>_tab.tex</code></li> </ul> <p>See <code>!TXTFORM</code> for more detail.</p>  |
| <code>!TXTFORM [n]</code>   | <p>sets the default argument for <code>!PVSFORM</code>, <code>!SLNFORM</code>, <code>!TABFORM</code> and <code>!YHTFORM</code> if these are not explicitly set.</p> <ul style="list-style-type: none"> <li>• <code>!TXTFORM</code> (or <code>!TXTFORM 1</code>) replaces multiple spaces with TAB and changes the file extension to, say, <code>_sln.txt</code>. This makes it easier to load the solutions into Excel.</li> <li>• <code>!TXTFORM 2</code> replaces multiple spaces with COMMA and changes the file extension to, say, <code>_sln.csv</code>. However, since factor labels sometimes contain commas, this form is not so convenient.</li> <li>• <code>!TXTFORM 3</code> replaces multiple spaces with Ampersand, appends a double backslash to each line and changes the file extension to say <code>_sln.tex</code> (Latex style).</li> </ul> <p>Additional significant digits are reported with these formats. Omitting the qualifier means the standard fixed field format is used. For <code>.yht</code> and <code>.sln</code> files, setting <code>n</code> to <code>-1</code> means the file is not formed.</p> |
| <code>!TWOWAY</code>        | 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.  |
| <code>!VCC n</code>         | this qualifier is deprecated. Its functionality has been replaced with the <code>VCC</code> directive used for linking variance parameters. Details are presented in Section 7.8.1.   |
| <code>!VGSECTORS [s]</code> | <p>requests that the variogram formed with radial coordinates be based on <code>s</code> (4, 6 or 8) sectors of size <math>180/s</math> degrees (see Section 2.4.2). The default is 4 sectors if <code>!VGSECTORS</code> is omitted and 6 sectors if it is specified without an argument. The first sector is centred on the <math>X</math> direction.</p> <p>Figure 5.1 is the variogram using radial coordinates obtained using predictors of random effects fitted as <code>fac(xsca,ysca)</code>. It shows low semi-variance in <code>xsca</code> direction, high semivariance in the <code>ysca</code> direction with intermediate values in the 45 and 135 degrees directions.</p>  |
| <code>!YHTFORM [f]</code>   | <p>controls the form of the <code>.yht</code> file</p> <ul style="list-style-type: none"> <li>• <code>!YHTFORM -1</code> suppresses formation of the <code>.yht</code> file</li> <li>• <code>!YHTFORM 1</code> is TAB separated: <code>.yht</code> becomes <code>_yht.txt</code></li> </ul>   |



### 5.8 Job control qualifiers

| qualifier | action   |
|-----------|--|
|           | <ul style="list-style-type: none"> <li>• !YHTFORM 2 is COMMA separated: .yht becomes _yht.csv</li> <li>• !YHTFORM 3 is Ampersand separated: .yht becomes _yht.tex</li> </ul> |
| !YSS [r]  | adds <i>r</i> to the total Sum of Squares. This might be used with !DF to add some variance to the analysis when analysing summarised data using weights.                    |

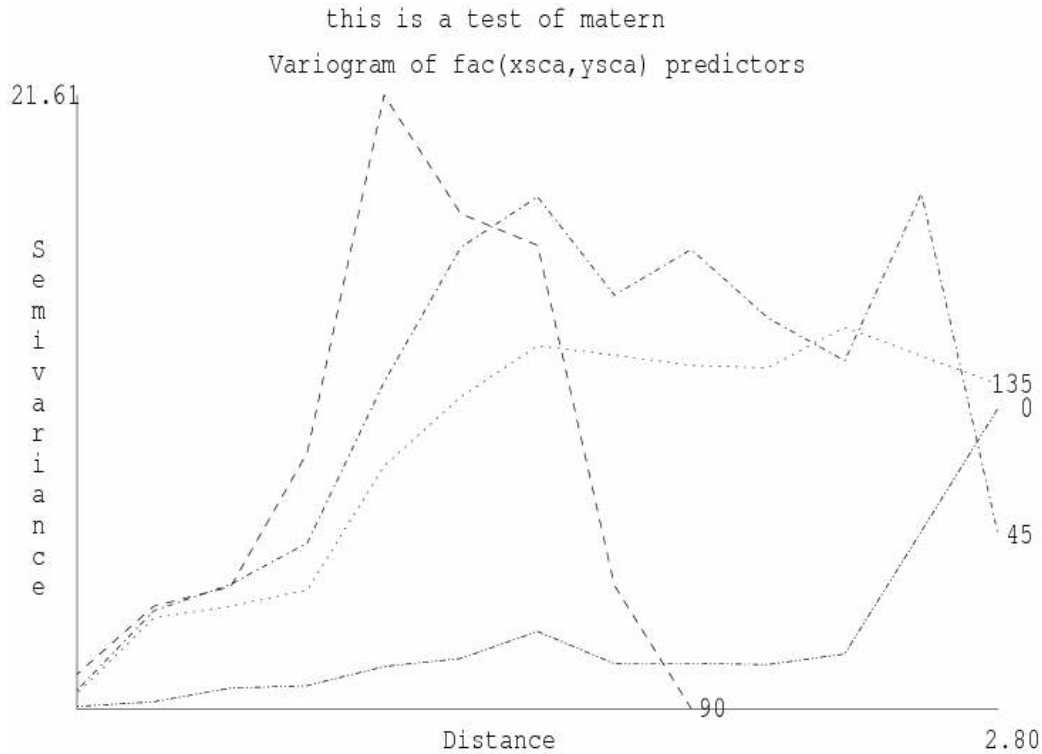


Figure 5.1: Variogram in 4 sectors for Cashmore data

Table 5.9: List of very rarely used job control qualifiers

| qualifier  | action   |
|--|--|
| !CINV [t]  | writes the known cells in the sparse portion of the $\mathbf{C}$ inverse, pertaining to model term <i>t</i> , to <i>basename.cii</i> . The known cells are primarily those that were not zero before inversion. The .cii file is ASCII sparse stored with <i>row column value</i> for each known cell, lower triangle row-wise.  |
| !CMAT,<br>!CPMAT,<br>!CAMAT,<br>!CIPMAT,<br>!CIMAT,<br>!DOUBLE | <p>these output matrices were added to facilitate research into efficient processing of the <math>\mathbf{C}</math> matrix of the mixed model equations. They write five forms of <math>\mathbf{C}</math> as follows</p> <ul style="list-style-type: none"> <li>• !CMAT writes <i>basename_Cmat.bin</i>, <math>\mathbf{C}</math> in model order</li> <li>• !CPMAT writes <i>basename_CPmat.bin</i>, <math>\mathbf{C}</math> permuted to analysis order</li> <li>• !CAMAT writes <i>basename_Camat.bin</i>, <math>\mathbf{C}</math> absorbed in analysis order</li> </ul> |

## 5.8 Job control qualifiers

| qualifier           | action   |
|---------------------|--|
|                     | <ul style="list-style-type: none"> <li>• !CIPMAT writes <i>basename_CIPmat.bin</i>, the sparse <math>\mathbf{C}^{-1}</math> in analysis order</li> <li>• !CIMAT writes <i>basename_CImat.bin</i>, the sparse <math>\mathbf{C}^{-1}</math> in model order</li> <li>• !DOUBLE stores the column numbers and matrix values in double precision, making the resulting file twice as large. The file extension is then <i>.dbin</i>.</li> </ul> <p>All forms are typically sparse and only the known non-zero cells are written. Note in particular that <b>ASReml</b> does not form those cells of <math>\mathbf{C}^{-1}</math> which are not present in the absorbed matrix since there are often many of them and they are not needed for the <b>AI REML</b> algorithm. The model order can be found in the <i>.asl</i> file (search there for SUBMODEL) and is the order in which fitted effects are reported in the <i>.sln</i> file. The first row/column actually pertains to the response variable cross-products. So called 'hidden equations' have been stripped out. The files can be read back with <b>FORTRAN</b> statements <i>inter alia</i> like</p> <pre> INTEGER :: IR, NR, IST, NV REAL, ALLOCATABLE : KV(:, :) ! DOUBLE PRECISION, ALLOCATABLE: KV(:, :) : OPEN(111, FILE="basename_Cmat.bin", FORM='UNFORMATTED') READ(111) NR ALLOCATE (KV(2, NR)) DO IR=1, NR READ(111) IST, NV, KV(IST:2, 1:NV) : ENDDO CLOSE(111) </pre> <p>The matrices are written row-wise. If the whole row is present, IST is 2, NV will equal IR and KV(2, 1:IR) will contain all (IR) values up to the diagonal. Otherwise, IST is 1, NV will be less than IR and specifies how many values are known, non-zero. KV(1, 1:NV) will contain the column numbers for the values in KV(2, 1:NV) in order with KV(1, NV) being equal to IR. Since these matrices are potentially huge, the values are written in single precision (REAL*4) by default, double precision (REAL*8) if !DOUBLE is specified.</p> <p>A second file, <i>basename_CaEqnOrder.bin</i>, holding the permuted order of the equations is also written with !CAMAT and !CPMAT.</p> <pre> INTEGER NR, EO(1:100000) OPEN(112, FILE="basename_CaEqnOrder.bin", FORM="UNFORMATTED") READ(112) NR, EO(1:NR) CLOSE(112) </pre> |
| !FACPOINTS <i>n</i> | affects the number of distinct points recognised by the <code>fac()</code> model function (Table 6.1). The default value of <i>n</i> is 1000 so that points closer than 0.1% of the range are regarded as the same point.  |
| !KNOTS <i>n</i>     | changes the default knot points used when fitting a spline to data with more than <i>n</i> different values of the spline variable. When there are more than <i>n</i> (default 50) points, <b>ASReml</b> will default to using <i>n</i> equally spaced knot points.  |
| !NOCHECK            | forces <b>ASReml</b> 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 <b>ASReml</b> to hold the data in memory. <b>ASReml</b> 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 <i>/tmp</i>   |

## 5.8 Job control qualifiers

| qualifier   | action  |
|---|---|
|   | directory which may not have enough space to hold it.   |
| <code>!POLPOINTS <i>n</i></code>                                      | affects the number of distinct points recognised by the <code>pol()</code> model function (Table 6.1). The default value of <i>n</i> is 1000 so that points closer than 0.1% of the range are regarded as the same point.   |
| <code>!PPOINTS <i>n</i></code>  | influences the number of points used when predicting splines and polynomials. The design matrix generated by the <code>leg()</code> , <code>pol()</code> and <code>spl()</code> functions are modified to include extra rows that are accessed by the <code>PREDICT</code> directive. The default value of <i>n</i> is 21 if there is no <code>!PPOINTS</code> qualifier. The range of the data is divided by <i>n</i> -1 to give a step size <i>i</i> . For each point <i>p</i> in the list, a predict point is inserted at <i>p</i> + <i>i</i> if there is no data value in the interval [ <i>p</i> , <i>p</i> + 1.1 × <i>i</i> ]. <code>!PPOINTS</code> is ignored if <code>!PVAL</code> is specified for the variable. This process also effects the number of levels identified by the <code>fac()</code> model term.  |
| <code>!REPORT</code>  | forces <b>ASReml</b> 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.   |
| <code>!SCALE 1</code>   | when forming a design matrix for the <code>spl()</code> model term, <b>ASReml</b> uses a standardized scale (independent of the actual scale of the variable). The qualifier <code>!SCALE 1</code> forces <b>ASReml</b> to use the scale of the variable. The default standardised scale is appropriate in most circumstances.  |
| <code>!SCORE</code>   | requests <b>ASReml</b> write the SCORE vector and the Average Information matrix to files <i>basename</i> .SCO and <i>basename</i> .AIM. The values written are from the last iteration.  |
| <code>!SLOW <i>n</i></code>   | reduces the update step sizes of the variance parameters more persistently than the <code>!STEP <i>r</i></code> qualifier. If specified, <b>ASReml</b> looks at the potential size of the updates and if any are large, it reduces the size of <i>r</i> . If <i>n</i> is greater than 10 <b>ASReml</b> also modifies the Information matrix by multiplying the diagonal elements by <i>n</i> . 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 <code>!SLOW</code> is set and <code>NIT</code> >1, <b>ASReml</b> immediately moves the variance parameters back towards the previous values and restarts the iteration.  |
| <code>!TOLERANCE [<i>s</i><sub>1</sub> [<i>s</i><sub>2</sub>]]</code> | <p>modifies the ability of <b>ASReml</b> to detect singularities in the mixed model equations. This is intended for use on the rare occasions when <b>ASReml</b> detects singularities after the first iteration; they are not expected.</p> <p>Normally (when no <code>!TOLERANCE</code> qualifier is specified), a singularity is declared if the adjusted sum of squares of a covariable is less than a small constant (<math>\eta</math>) or less than the uncorrected sum of squares <math>\times \eta</math>, where <math>\eta</math> is <math>10^{-8}</math> in the first iteration and <math>10^{-10}</math> thereafter. The qualifier scales <math>\eta</math> by <math>10^5</math> for 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.</p> <p>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.</p> |
| <code>!VRB</code>   | contains the estimates of the fixed effects and their variance approximate prediction variance matrix corresponding to the dense portion of the variance.   |
| <code>!WVR</code>   | reports working variables to a <code>.wvwr</code> 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

$$\mathbf{y} = \mathbf{X}\boldsymbol{\tau} + \mathbf{Z}\mathbf{u} + \mathbf{e} \quad (6.1)$$

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

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

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.

```
NIN Alliance Trial 1989
  variety
  :
  column 11
nin89.asd !SKIP 1
yield ~ mu variety !r idv(repl) !f mv
residual idv(units)
```

The syntax for specifying the model is

```
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](#).
- *qualifiers* 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 their variance structures. If no variance structure is formally given, often `idv()` is assumed, 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 `~` ('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 (or a comma) must appear on each of the incomplete lines 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 can lose clarity, ambiguity and confusion. For example, if the truncated form matches more than one label, the term associated with the first match is assumed, for example, `dens` is an abbreviation for `density` but `spl(dens, 7)` is a different model term to `spl(density, 7)` because it does not represent a simple truncation of the label.

## 6.2 Specifying model formulae in ASReml

- 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.

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.

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)
```

In Table 6.1 and Table 6.3, 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.

Table 6.1: Summary of reserved words, operators and functions

| model term     | brief description  | common usage |        |
|----------------|--|--------------|--------|
|                |  | fixed        | random |
| reserved terms |  |              |        |
| half           | used in the linear model to include a column with coefficient of 0.5. It is primarily used to scale interacting terms as in <code>half.sire</code> and <code>half.dam</code> . |              | ✓      |
| mu             | the constant term or intercept.  | ✓            |        |
| mv             | a term to estimate missing values.   | ✓            |        |
| Trait          | multivariate counterpart to <code>mu</code> .  | ✓            |        |
| units          | forms a factor with a level for each experimental unit.  |              | ✓      |
| zero           | the constant 0 (constructed like <code>mu</code> , but with coefficients of 0).  |              |        |
| 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.  | ✓            | ✓      |

## 6.2 Specifying model formulae in ASReml

| model term                     | brief description   | common usage |        |
|--------------------------------|---|--------------|--------|
|                                |   | fixed        | random |
| ,                              | 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).   |              | ✓      |
| <b>commonly used functions</b> |   |              |        |
| at( $f, n$ )                   | condition on level $n$ of factor $f$ .<br>$n$ 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 variate 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 $v$ .   |              | ✓      |
| <b>other functions</b>         |   |              |        |
| $t\{n\}$                       | 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. If $t$ is complex it may be necessary to predefine it by saying - $t$ and( $t$ , $r$ ).    |              | ✓      |
| c( $f$ )                       | factor $f$ is fitted with <i>sum to zero</i> constraints.   | ✓            |        |
| cos( $v, r$ )                  | forms cosine from $v$ with period $r$ .   | ✓            |        |
| ge( $f$ )                      | condition on factor/variable $f \geq r$ .   | ✓            |        |
| gt( $f$ )                      | condition on factor/variable $f > r$ .  | ✓            |        |
| h( $f$ )                       | factor $f$ is fitted <i>Helmert</i> constraints.  | ✓            |        |
| hs( $f, k$ )                   | is a factor with level 1 if factor $f$ is coded 1 or $k$ and zero otherwise.  | ✓            | ✓      |
| inv( $v$ [, $r$ ])             | forms reciprocal of $v + r$ .   | ✓            |        |
| le( $f$ )                      | condition on factor/variable $f \leq r$ .   | ✓            |        |
| 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                            | forms an MA1 design matrix from plot numbers.   |              | ✓      |

## 6.2 Specifying model formulae in ASReml

| model term                                | brief description  | common usage |        |
|---|--|--------------|--------|
|   |  | fixed        | random |
| <code>mbf(v, r)</code>                    | is a factor derived from data factor <code>v</code> by using the <code>!MBF</code> qualifier.  | ✓            | ✓      |
| <code>out(n)</code>                       | condition on observation <code>n</code> .  | ✓            |        |
| <code>out(n, t)</code>                    | condition on record <code>n</code> , trait <code>t</code> .  | ✓            |        |
| <code>pol(v, [-] n)</code>                | forms $n+1$ orthogonal polynomials of order 0 (intercept), 1 (linear). .<br>. <code>n</code> from the values in <code>v</code> ; the intercept polynomial is omitted if <code>n</code> is preceded by the negative sign.   | ✓            |        |
| <code>pow(x, p[, o])</code>               | defines the covariable $(x + o)^p$ for use in the model where <code>x</code> is a variable in the data, <code>p</code> is a power and <code>o</code> is an offset.   | ✓            |        |
| <code>qtl(f, p)</code>                    | impute a covariable from marker map information at position <code>p</code> .   | ✓            |        |
| <code>ref(f, k)</code>                    | creates a factor using levels from <code>f</code> except that level <code>k</code> is set to zero. This effectively sets <code>k</code> (default 1) as the reference level of the factor <code>f</code> when <code>mu</code> and <code>ref(f, k)</code> are fitted and the level <code>j</code> effect of <code>ref(f, k)</code> estimates the difference of level <code>j</code> and <code>k</code> of the factor <code>f</code> effects. | ✓            |        |
| <code>sin(v, r)</code>                    | forms sine from <code>v</code> with period <code>r</code> .  | ✓            |        |
| <code>sqrt(v[, r])</code>                 | forms square root of <code>v + r</code> .  | ✓            |        |
| <code>uni(f)</code>                       | forms a factor with a level for each record where factor <code>f</code> is non-zero.   |              | ✓      |
| <code>vect(v)</code>                      | is used in a multivariate analysis on a multivariate set of covariates ( <code>v</code> ) to pair them with the variates.  | ✓            | ✓      |
| <code>zero(k)</code><br><code>zero</code> | is used to include <code>k</code> column(s) of zeros in the model. It is primarily intended to be used to set aside space for fitting the factors of an extended factor analytic (XFA <code>k</code> ) (or reduced rank, RR <code>k</code> ) model applied across several model terms. For example:<br><br><code>str(ani age.ani zero(1).ani xfal(2).ani)</code><br><br>zero is equivalent to <code>zero(1)</code> .                       |              | ✓      |

### 6.2.2 Examples

Table 6.2 provides with some examples of the use of ASReml model formulae.

Table 6.2: Simple examples of model formulae

| ASReml code   | action   |
|---|--|
| <code>yield ~ mu variety</code><br><code>residual idv(units)</code>                           | fits a model with a constant and fixed variety effects.  |
| <code>yield ~ mu variety !r idv(block)</code><br><code>residual idv(units)</code>             | fits a model with a constant term, fixed variety effects and random block effects.                       |
| <code>yield ~ mu time variety time.variety</code><br><code>residual idv(units)</code>         | fits a saturated model with fixed time and variety main effects and time by variety interaction effects. |
| <code>livewt ~ mu breed sex breed.sex !r idv(sire)</code><br><code>residual idv(units)</code> | 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.
- The reserved word `mv` is used for fitting missing values and its role and placement are presented in Table 6.3.

```
NIN Alliance Trial 1989
  variety !A
  :
  row 22
  column 11
nin89.asd !SKIP 1 !MVINCLUDE
yield ~ mu variety mv !r idv(repl)
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.
- When missing values are to be estimated, `mv` can be placed as the last *fixed* term or in *sparse\_fixed* and is always fitted as part of *sparse\_fixed*.

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

In [Chapter 7](#) we discuss 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 6.3), 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 6.3).
- 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 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` and

`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. `diag(site).geno`) which associates 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 Section 10.3.4). 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 design functions

Table 6.3 presents detailed descriptions of the model design functions discussed above. Note that some three letter function names may be abbreviated to the first letter.

Table 6.3: Alphabetic list of model design functions and descriptions

| model function           | action   |
|--------------------------|--|
| <code>abs(v)</code>      | takes the absolute value of the variable $v$ . This function can be used on the response variable.   |
| <code>and(t, r)</code>   | 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   |
| <code>a(t, r)</code>     | <code>male and(female)</code><br>assuming the $i$ th male is the same individual as the $i$ th female.   |
| <code>at(f, n)</code>    | 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  |
| <code>@(f, n)</code>     | <code>at(site, 3).row</code><br>The string <code>@(</code> is equivalent to <code>at(</code> for this function.  |
| <code>at(f)</code>       | <code>at(f)</code> is expanded to a series of terms like <code>at(f, i)</code> 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.   |
| <code>@(f)</code>        |  |
| <code>at(f, m, n)</code> | <code>at(f, i, j, k)</code> is expanded to a series of terms <code>at(f, i) at(f, j) at(f, k)</code> . Similarly, <code>at(f, i).X at(f, j).X at(f, k).X</code> can be written as <code>at(f, i, j, k).X</code> provided   |
| <code>@(f, m, n)</code>  | <code>at(f, i, j, k)</code> is written as the first component of the interaction. Any number of levels may be listed. Contiguous sets of values can be specified as $i:j$ .  |
| <code>cos(v, r)</code>   | forms cosine from $v$ with period $r$ . Omit $r$ if $v$ is radians. If $v$ is degrees, $r$ is 360.   |
| <code>con(f)</code>      | apply <i>sum to zero</i> constraints to factor $f$ . It is not appropriate for random factors and fixed factors with missing cells. <b>ASReml</b> assumes you specify the correct number of levels for each factor. The formal effect of the <code>con()</code> function is to form a model term with the highest level formally equal to minus the sum of the preceding terms. With <i>sum to zero</i> 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. |
| <code>c(f)</code>        |  |
| <code>fac(v)</code>      | <code>fac(v)</code> forms a factor with a level for each value of $x$ and any additional points inserted as discussed with the qualifiers <code>!PPOINTS</code> and <code>!PVAL</code> . <code>fac(v,y)</code> forms a factor with a level for each combination of values from $v$ and $y$ . The values are reported in the <code>.res</code> file.  |
| <code>fac(v, y)</code>   |  |

## 6.6 Alphabetic list of model design functions

| model function            | action  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |    |   |    |    |    |   |   |   |   |    |   |   |    |    |   |   |   |   |    |   |   |   |    |   |   |   |   |    |   |   |   |   |    |    |    |    |
|---------------------------|---|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|----|---|----|----|----|---|---|---|---|----|---|---|----|----|---|---|---|---|----|---|---|---|----|---|---|---|---|----|---|---|---|---|----|----|----|----|
| <code>h(f)</code>         | <p><code>h(f)</code> requests <b>ASReml</b> to fit the model term for factor <i>f</i> using Helmert constraints. Neither Sum-to-zero nor Helmert constraints generate interpretable effects if singularities occur. <b>ASReml</b> 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.</p> <table><thead><tr><th></th><th>H1</th><th>H2</th><th>H3</th><th>H4</th><th>C1</th><th>C2</th><th>C3</th><th>C4</th></tr></thead><tbody><tr><td>F1</td><td>-1</td><td>-1</td><td>-1</td><td>-1</td><td>1</td><td>0</td><td>0</td><td>0</td></tr><tr><td>F2</td><td>1</td><td>-1</td><td>-1</td><td>-1</td><td>0</td><td>1</td><td>0</td><td>0</td></tr><tr><td>F3</td><td>0</td><td>2</td><td>-1</td><td>-1</td><td>0</td><td>0</td><td>1</td><td>0</td></tr><tr><td>F4</td><td>0</td><td>0</td><td>3</td><td>-1</td><td>0</td><td>0</td><td>0</td><td>1</td></tr><tr><td>F5</td><td>0</td><td>0</td><td>0</td><td>4</td><td>-1</td><td>-1</td><td>-1</td><td>-1</td></tr></tbody></table> |    | 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 |
|                           | 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 |    |    |    |    |    |    |    |   |   |   |   |    |   |    |    |    |   |   |   |   |    |   |   |    |    |   |   |   |   |    |   |   |   |    |   |   |   |   |    |   |   |   |   |    |    |    |    |
| <code>ide(f)</code>       | is used to take a copy of a pedigree factor <i>f</i> and fit it without the genetic relationship covariance. This facilitates fitting a <i>second animal effect</i> . 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, namely   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |    |   |    |    |    |   |   |   |   |    |   |   |    |    |   |   |   |   |    |   |   |   |    |   |   |   |   |    |   |   |   |   |    |    |    |    |
| <code>i(f)</code>         | <pre>!r !{ animal dam !} ide(dam)</pre>   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |    |   |    |    |    |   |   |   |   |    |   |   |    |    |   |   |   |   |    |   |   |   |    |   |   |   |   |    |   |   |   |   |    |    |    |    |
| <code>inv(v[,r])</code>   | forms the reciprocal of $v + r$ . This may also be used to transform the response variable.   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |    |   |    |    |    |   |   |   |   |    |   |   |    |    |   |   |   |   |    |   |   |   |    |   |   |   |   |    |   |   |   |   |    |    |    |    |
| <code>leg(v, [-]n)</code> | forms <i>n</i> +1 Legendre polynomials of order 0 (intercept), 1 (linear), . . . <i>n</i> from the values in <i>v</i> ; the intercept polynomial is omitted if <i>n</i> is preceded by the negative sign. The actual values of the coefficients are written to the <code>.res</code> file. This is similar to the <code>pol()</code> function described below.  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |    |   |    |    |    |   |   |   |   |    |   |   |    |    |   |   |   |   |    |   |   |   |    |   |   |   |   |    |   |   |   |   |    |    |    |    |
| <code>lin(f)</code>       | takes the coding of factor <i>f</i> as a covariate. The function is defined for <i>f</i> being a simple factor, Trait and units. The <code>lin(f)</code> function does not centre or scale the variable.  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |    |   |    |    |    |   |   |   |   |    |   |   |    |    |   |   |   |   |    |   |   |   |    |   |   |   |   |    |   |   |   |   |    |    |    |    |
| <code>l(f)</code>         | <b>Motivation:</b> Sometimes you may wish to fit a covariate as a random factor as well. If the coding is say <i>1 . . n</i> , then you should define the field as a factor in the field definition and use the <code>lin()</code> 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 <code>fac()</code> function to derive a factor version.  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |    |   |    |    |    |   |   |   |   |    |   |   |    |    |   |   |   |   |    |   |   |   |    |   |   |   |   |    |   |   |   |   |    |    |    |    |
| <code>log(v[,r])</code>   | forms the natural log of $v + r$ . This may also be used to transform the response variable.  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |    |   |    |    |    |   |   |   |   |    |   |   |    |    |   |   |   |   |    |   |   |   |    |   |   |   |   |    |   |   |   |   |    |    |    |    |
| <code>ma1</code>          | creates a first-differenced design matrix which, when defining a random effect, is equivalent to fitting a moving average variance structure in one dimension. The first-difference operator is coded across all data points (assuming they are in time/space order). Previous releases suggested that the form <code>ma1(factor)</code> could be used but this caused confusion with the variance function <code>ma1(factor)</code> , introduced in Release 3, for fitting moving average models. In Release 4.2 <code>ma1(factor)</code> is interpreted as a variance function.   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |    |   |    |    |    |   |   |   |   |    |   |   |    |    |   |   |   |   |    |   |   |   |    |   |   |   |   |    |   |   |   |   |    |    |    |    |
| <code>mbf(f, c)</code>    | is a term that is predefined by using the <code>!MBF</code> qualifier (Table 5.7).  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |    |   |    |    |    |   |   |   |   |    |   |   |    |    |   |   |   |   |    |   |   |   |    |   |   |   |   |    |   |   |   |   |    |    |    |    |
| <code>mbf(f)</code>       |   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |    |   |    |    |    |   |   |   |   |    |   |   |    |    |   |   |   |   |    |   |   |   |    |   |   |   |   |    |   |   |   |   |    |    |    |    |
| <code>mu</code>           | 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.  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |    |   |    |    |    |   |   |   |   |    |   |   |    |    |   |   |   |   |    |   |   |   |    |   |   |   |   |    |   |   |   |   |    |    |    |    |

## 6.6 Alphabetic list of model design functions

| model function                                       | action   |
|--|--|
| <code>mv</code>                                      | <p>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 <code>mv</code> and any terms that follow are always included in the sparse set of equations.</p> <p>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 <b>ASReml</b> 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 <code>mv</code> in the model.</p>   |
| <code>out(n)</code><br><code>out(n, t)</code>        | <p><code>out(n)</code>, <code>out(n, t)</code> establishes a binary variable which is</p> <ul style="list-style-type: none"> <li>• <code>out(i)</code> 1 if data relates to observation <math>i</math> (trait 1), else is 0.</li> <li>• <code>out(i, t)</code> 1 if data relates to observation <math>i</math> (trait <math>t</math>), else is 0.</li> </ul> <p>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 <code>.res</code> file) and the appropriate record numbers for the <code>out()</code> term are reported in the <code>.res</code> file. Note that <math>i</math> 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 <code>!D</code> transformation in association with the <code>!=V0</code> transformation.)</p> |
| <code>pol(v, [-]n)</code><br><code>p(v, [-]n)</code> | <p>forms a set of orthogonal polynomials of order <math> n </math> based on the unique values in variate (or factor) <math>v</math> and any additional interpolated points, see <code>!PPOINTS</code> and <code>!PVAL</code> in Table 5.7. It includes the intercept if <math>n</math> is positive, omits it if <math>n</math> is negative. For example</p> <pre>pol(time, 2)</pre> <p>forms a design matrix with three columns of the orthogonal polynomial of degree 2 from the variable <code>time</code>. Alternatively</p> <pre>pol(time, -2)</pre> <p>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.</p> <p>The actual values (Robson, 1959, Steep and Torrie, 1960) of the coefficients are written to the <code>.res</code> file. This factor could be interacted with a design factor to fit random regression models. The <code>leg()</code> function differs from the <code>pol()</code> function in the way the quadratic and higher polynomials are calculated.</p>   |
| <code>pow(x, p[, o])</code>                          | <p>defines the covariable <math>(x + o)^p</math> for use in the model where <math>x</math> is a variable in the data, <math>p</math> is a power and <math>o</math> is an offset.</p> <pre>pow(x, 0.5[, o])</pre> <p>is equivalent to</p> <pre>sqr(x[, o])</pre> <pre>pow(x, 0[, o])</pre> <p>is equivalent to</p> <pre>log(x[, o])</pre> <pre>pow(x, -1[, o])</pre> <p>is equivalent to</p> <pre>inv(x[, o])</pre>   |

## 6.6 Alphabetic list of model design functions

| model function                                   | action  |
|--|---|
| <code>qtl(f, r)</code>                           | calculates an expected marker state from flanking marker information at position $r$ of the linkage group $f$ (see <code>!MM</code> to define marker locations). $r$ may be specified as <code>\$TPn</code> where <code>\$TPn</code> has been previously internally defined with a <code>PREDICT</code> statement (see <code>!TURNINGPOINTS</code> in Table 10.2). $r$ should be given in Morgans.  |
| <code>sin(v, r)</code>                           | forms sine from $v$ with period $r$ . Omit $r$ if $v$ is radians. If $v$ is degrees, $r$ is 360.  |
| <code>s(v[,k])</code><br><code>spl(v[,k])</code> | In order to fit spline models associated with a variate $v$ and $k$ knot points in <b>ASReml</b> , $v$ is included as a covariate in the model and <code>spl(v,k)</code> as a random term. The knot points can be explicitly specified using the <code>!SPLINE</code> qualifier (Table 5.7). If $k$ is specified but <code>!SPLINE</code> 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 <code>.res</code> file. An example of the use of <code>spl()</code> is<br><br><code>price ~ mu week !r spl(week)</code>  |
| <code>sqrt(v[,r])</code>                         | forms the square root of $v + r$ . This may also be used to transform the response variable.  |
| <code>Trait</code>                               | is used with multivariate data to fit the individual trait means. It is formally equivalent to <code>mu</code> but <code>Trait</code> is a more natural label for use with multivariate data. It is interacted with other factors to estimate their effects for all traits.   |
| <code>units</code>                               | 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.   |
| <code>uni(f[,0[,n]]new)</code>                   | 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.   |
| <code>uni(f,k[,n])</code>                        | creates a factor with a level for every record subject to the factor level of $f$ equalling $k$ , <i>i.e.</i> a level is created for the factor whenever a record is encountered whose integer truncated data value from data field $f$ is $k$ . Thus<br><br><code>uni(site,2)</code><br><br>would be used to create an independent error term for site 2 in a multi-environment trial and is equivalent to<br><br><code>at(site,2).units</code><br><br>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 <code>.sln</code> file may be substantially larger than needed. However, if the <code>units</code> vector is full size, the effects are mapped by record number and added back to the fitted residual for creating 'residual' plots.   |
| <code>vect(v)</code>                             | is used in a multivariate analysis on a multivariate set of covariates ( $v$ ) to pair them with the variates. The test example included<br><br><code>signal !G 93 # 93 slides</code><br><code>background !G 93</code><br><code>dart.asd !ASUV</code><br><code>signal ~ Trait Trait.vect(background) ...</code><br><br>to fit a slide specific regression of <code>signal</code> on <code>background</code> . In this example, <code>signal</code> is a multivariate set of 93 variates and <code>background</code> is a set of 93 covariates. The <code>signal</code> values relate to either the Red or Green channels. So for each slide and channel, we need to fit a simple regression of <code>signal ~ mu background</code> . But the data for the 93 slides is presented in parallel. If it were presented in series, with a factor <code>slide</code> indexing the slides, the equivalent model would be<br><br><code>signal~ slide slide.background.</code> |

## 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 some factor, then the residual line can be `residual idv(units)` to estimate this scaling factor, or `residual idv(units !INIT v !GF)` to set this scaling factor at  $v$ . If `wt` are true weights (inverse known residual variances), the default, these can be specified explicitly on the residual line as `residual idv(units !INIT 1 !GF)`, or `residual id(units)`, or `residual units`. Note that this default and specification has changed from previous releases in order to rationalize the specification of scaling factors in weights and dispersion factors in GLMM.

Weights can also be used when entering summarized data in which case the `!DF` qualifier (Table 5.8) must be used to adjust the degrees of freedom, and the `!YSS` qualifier may be needed to adjust the sampling variance.

When a structure is present in the residuals (Section 7.3) the weights are applied as a matrix product. If  $\Sigma$  is the structure and  $W$  is the diagonal matrix constructed from the square root of the values of the weight variate, then  $R^{-1} = W\Sigma^{-1}W$ . 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

$y$  — the observation

$n$  — the count for grouped data specified by the `!TOTAL` qualifier

$\phi$  — a parameter set with the `!PHI` qualifier

$\mu$  — the mean on the data scale calculated using the inverse link function from the predicted value (linear predictor)  $\eta$  on the underlying scale where  $\eta = X\tau$

$v$  — the variance under some distributional assumption calculated as a function of  $\mu$  and  $n$

$d$  — the deviance (-twice the log-likelihood) for that distribution.

Note that  $\mu$  is the mean on the data scale, and  $\eta = X\tau$  is the linear predictor on the underlying scale.



Table 6.4: Link qualifiers and functions

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

GLMs are specified by qualifiers after the name of the dependent variable but before the  $\sim$  character. Table 6.4 lists the link function qualifiers which relate the linear predictor ( $\eta$ ) scale to the observation ( $\mu = E[y]$ ) scale. Table 6.5 lists the distribution and other qualifiers.

Table 6.5: GLM distribution qualifiers

The default link is listed first followed by permitted alternatives.

| qualifiers   | action   |
|--|--|
| !NORMAL [ !IDENTITY   !LOGARITHM   !INVERSE ]  | allows the model to be fitted on the log/inverse scale but with the residuals on the natural scale. !NORMAL !IDENTITY is the default.  |
| !BINOMIAL [ !LOGIT   !IDENTITY   !PROBIT   COMPLOGLOG ] [ !TOTAL $n$ ]   | 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). For this distribution, using the qualifier !WT $n$ to carry out a weighted analysis and the qualifier !TOTAL $n$ are synonyms. If the user wishes to use a weight, $w$ , other than the total, $n$ , the $y$ variable should be presented as a proportion to avoid transforming $y$ by dividing by $w$ . 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. |
| $v = \mu(1 - \mu)/n$ $d = 2n(y \ln(y/\mu) + (1 - y) \ln(\frac{1-y}{1-\mu}))$   |  |
| !MULTINOMIAL $k$ !CUMULATIVE [ !LOGIT   !PROBIT   COMPLOGLOG ] [ !TOTAL $n$ ]  | fits a multiple threshold model with $t = k - 1$ thresholds to polytomous ordinal data with $k$ categories assuming a multinomial distribution.  |
| $v_{ij} = \mu_i(1 - \mu_j)/n$ for $i \leq j \leq t$ $d = 2n \sum_{i=1}^k (y_i \ln(y_i/p_i))$ where $Y_i = \sum_{j=1}^i y_j$ $\mu_i = E(Y_i) \text{ and}$ | 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 $k$ th category) must be given in another variable indicated by the !TOTAL qualifier: the multinomial model requires a particular variance structure across the multinomial classes.   |

## 6.8 Generalized Linear (Mixed) Models

| qualifiers   | action   |
|--|--|
| $p_i = \mu_i - \mu_{i-1}$  | <p>This is formally specified as</p> <pre>residual id(units).mthr(Trait)</pre> <p>The multinomial threshold model is fitted as a cumulative probability model. The proportions (<math>y_i = r_i/n</math>) in the ordered categories are summed to form the cumulative proportions (<math>Y_i</math>) which are modelled with logit (!LOGIT), probit (!PROBIT) or Complementary LogLog (!CLOG) link functions. The implicit residual variance on the underlying scale is <math>\pi^2/3 \sim 3.3</math> (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 <math>\pi^2/6 \sim 1.65</math>. For example</p> <pre>lodging !MULTINOMIAL 4 !CUMULATIVE ~ Trait variety !r block PREDICT variety</pre> <p>where <code>Lodging</code> is a variate of ordered lodging scores, or a factor of ordered categories (if the factor is specified as names with !A or !I then the user may need to use !SORT or !L to order the levels appropriately, see Section 5.4.3). Predicted values are reported for the cumulative proportions.</p> |
| <pre>!POISSON [ !LOGARITHM   !IDENTITY   !SQRT ]</pre> $v = \mu$ $d = 2(y \ln(y/\mu) - (y - \mu))$   | <p>Natural logarithms are the default link function.</p> <p><b>ASReml</b> assumes the Poisson variable is not negative.</p>  |
| <pre>!GAMMA [ !INVERSE   !IDENTITY   !LOGARITHM ] [ !PHI <math>\phi</math> ] [ !TOTAL <math>n</math> ]</pre> $v = \mu^2/(\phi n)$ $d = 2n(-\phi \ln(\frac{\phi y}{\mu}) + \frac{\phi y - \mu}{\mu})$ | <p>The inverse is the default link function. <math>n</math> is defined with the !TOTAL qualifier and would be degrees of freedom in the typical application to mean-squares. The default value of <math>\phi</math> is 1.</p>  |
| <pre>!NEGBIN [ !LOGARITHM   !IDENTITY   !INVERSE ] [ !PHI <math>\phi</math> ]</pre> $v = \mu + \mu^2/\phi$ $d = 2((\phi + y) \ln(\frac{\mu + \phi}{y + \phi}) + y \ln(\frac{y}{\mu}))$               | <p>fits the Negative Binomial distribution. Natural logarithms are the default link function. The default value of <math>\phi</math> is 1.</p>   |
| <b>general qualifiers</b>  |  |
| !AOD   | <p>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 of its use is</p> <pre>LS !BIN !TOT COUNT !AOD ~ mu SEX GROUP</pre> <p>!AOD may not be used in association with PREDICT.</p>   |
| !DISP [ $h$ ]  | <p>this qualifier is deprecated (its functionality can be specified using the residual line). It was considered to specify an <i>overdispersion</i> scaling parameter (<math>h</math>) in the weights. <b>ASReml</b> estimates it as the residual variance of the working variable. Traditionally it is estimated from the deviance residuals, reported by <b>ASReml</b> as Variance heterogeneity.</p> <p>The default option is setting the dispersion factor to 1 and this will occur when using</p> <pre>residual idv(units 1 !GF) # Dispersion factor fixed to 1</pre>   |

## 6.8 Generalized Linear (Mixed) Models

| qualifiers                 | action  |
|----------------------------|---|
|                            | <p>or</p> <pre>residual id(units)          # Dispersion factor fixed to 1</pre> <p>or when omitting the residual line. Dispersion options can be set using</p> <pre>residual idv(units)         # Dispersion factor is estimated residual idv(units h !GF)   # Dispersion factor is set to h</pre>  |
| <code>!OFFSET [o]</code>   | <p>is used especially with binomial data to include an offset in the model where <i>o</i> 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</p> <pre>count !POIS !OFFSET base ~ mu group</pre> <p>The offset is included in the model as <math>\boldsymbol{\eta} = \mathbf{X}\boldsymbol{\tau} + o</math>. The offset will often be something like <math>\ln(n)</math>.</p> |
| <code>!TOTAL [n]</code>    | <p>is used especially with binomial and ordinal data where <i>n</i> is the field containing the total counts for each sample. If omitted, count is taken as 1.</p>  |
| <b>residual qualifiers</b> | <p>control the form of the residuals returned in the <code>.yht</code> file. The predicted values returned in the <code>.yht</code> file will be on the linear predictor scale if the <code>!WORK</code> or <code>!PVW</code> qualifiers are used. They will be on the observation scale if the <code>!DEVIANC</code>, <code>!PEARSON</code>, <code>!RESPONSE</code> or <code>!PVR</code> qualifiers are used.</p>  |
| <code>!DEVIANC</code>      | <p>produces deviance residuals, the signed square root of <math>d/h</math> from Table 6.5 where <i>h</i> is the dispersion parameter controlled by the <code>residual</code> statement. This is the default.</p>  |
| <code>!PEARSON</code>      | <p>writes Pearson residuals, <math>\frac{y-\mu}{\sqrt{v}}</math>, in the <code>.yht</code> file.</p>  |
| <code>!PVR</code>          | <p>writes fitted values on the response scale in the <code>.yht</code> file. This is the default.</p>   |
| <code>!PVW</code>          | <p>writes fitted values on the linear predictor scale in the <code>.yht</code> file.</p>  |
| <code>!RESPONSE</code>     | <p>produces simple residuals, <math>y - \mu</math>.</p>   |
| <code>!WORK</code>         | <p>produces residuals on the linear predictor scale, <math>\frac{y-\mu}{d\mu/d\boldsymbol{\eta}}</math>.</p>  |

**ASReml 4** allowed a bivariate analysis of a binomially distributed variate and a linear model variate (normally distributed variate with an identity link). **ASReml 4.2** has extended this bivariate analysis of generalized linear models. Both variates are now allowed to be distributed with Normal, Binomial, Poisson, Gamma or Negative Binomial distributions. Because multinomial threshold models are an extension of GLM with composite link functions they may not be included in a bivariate analysis. The GLMM qualifiers related to a variate should be specified immediately after the response variate is nominated. If a link function is specified, it must follow the distribution qualifier. Examples include

```
Scald !BINOMIAL !PROBIT FootRot !BINOMIAL !PROBIT ~ Trait ...
Scald !BINOMIAL !PROBIT weight ~ Trait ...
```

Although **ASReml** will provide an analysis of grouped binomial data it is usually more statistically efficient, if possible, to expand to the binary data and get a more efficient estimate of the residual covariance. The procedure used is to scale the specified residual variance matrix by the inverse

GLM weights, so the values in the specified residual variance matrix are actually dispersion factors, typically initialized with variances of 1.0. For instance

```
residual units.us(Trait !INIT 1 0.5 1)
```

or to fix dispersion

```
residual units.us(Trait !INIT 1 0.5 1 !GFPF)
```

The algorithm used is a heuristic extension to the standard PQL method used in the univariate case.

### 6.8.1 Generalized Linear Mixed Models

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 Schall's technique (Schall, 1991), pseudo-likelihood (Wolfinger and OConnell, 1993) and joint maximization (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 *et al.*, 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 re-estimated 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 !A
:
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 5.3.

**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. This 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 5.8. 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 typically singular. However, the fact that singularities occurred will be noted at the end of the `.asr` file. A likelihood ratio test is not valid if the fixed model has changed.

## 6.10.4 Examples of aliasing

The sequence of models in Table 6.6 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 6.6: Examples of aliasing in ASReml

| model   | number of singularities | order of fitting   |
|---|-------------------------|--|
| <code>yield ~ var !r idv(rep)</code>                    | 0                       | rep var  |
| <code>yield ~ mu var !r idv(rep)</code>                 | 1                       | rep mu var<br>first level of <code>var</code> is aliased and set to zero.  |
| <code>yield ~ var trt !r idv(rep)</code>                | 1                       | rep var trt<br>var fully fitted, first level of <code>trt</code> is aliased and set to zero.   |
| <code>yield ~ mu var trt var.trt, !r idv(rep)</code>    | 8                       | rep mu var trt var.trt<br>first levels of both <code>var</code> and <code>trt</code> are aliased and set to zero, together with subsequent interactions.   |
| <code>yield ~ mu var trt !r idv(rep), !f var.trt</code> | 8                       | [var.trt rep] mu var trt<br>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 (Table 5.6). Use `!DDF -1` to prevent the calculation to save processing time when significance testing is not required.



## 7 Specifying the variance structures

In [Chapter 2](#) we presented the general linear mixed model

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

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

$$\begin{bmatrix} \mathbf{u} \\ \mathbf{e} \end{bmatrix} \sim N \left( \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \end{bmatrix}, \begin{bmatrix} \mathbf{G}(\sigma_g) & \mathbf{0} \\ \mathbf{0} & \mathbf{R}_v(\sigma_r) \end{bmatrix} \right)$$

where the matrices  $\mathbf{G}$  and  $\mathbf{R}_v$  are variance matrices for  $\mathbf{u}$  and  $\mathbf{e}$  and are functions of parameters  $\sigma_g$  and  $\sigma_r$ . Under this parameterization

$$\text{var}(\mathbf{y}) = \mathbf{Z}\mathbf{G}(\sigma_g)\mathbf{Z}^\top + \mathbf{R}_v(\sigma_r)$$

- $\mathbf{G}$  structures for the random model terms (Section 2.1.3) and  $\mathbf{R}$  structures for the residual error term (Section 0)
- Direct sum structures for  $\mathbf{G}$  and/or  $\mathbf{R}_v$  ( $\mathbf{R}_c$ , 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)

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

where  $\gamma_g$  and  $\gamma_r$  represent the variance structure parameters associated with scaled (by  $\sigma_e^2$ ) variance matrices. Under this parameterization

$$\text{var}(\mathbf{y}) = \sigma_e^2 (\mathbf{Z}\mathbf{G}(\gamma_g)\mathbf{Z}^\top + \mathbf{R}_c(\gamma_r)).$$

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  $\mathbf{u}_i$  in  $\mathbf{u}$  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 7.10.

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` (e.g. `idv()`) to add a common variance, *i.e.* same variance across all rows, or with `h` (e.g. `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 ( $\mathbf{Z}_i$ ) and variance model ( $\mathbf{G}_i$ ), in particular allowing  $\mathbf{G}_i$  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 models are classified as *scaled* variance models if they include a scale parameter, or as *unscaled* variance models if their scale is fixed. Unscaled models include correlation models (diagonal elements equal to 1) and models generated by relationship matrices where the covariance structure is known except for the scale.

Table 7.1: List of common variance model functions

Their type (scaled or unscaled), the form of the variance matrix generated ( $\mathbf{V}$  for scaled variance matrix,  $\mathbf{U}$  for unscaled variance matrix), and a brief description. Parameters  $\sigma_i^2 > 0$  are variances,  $-1 < \rho_i < 1$  are correlations. Subscript  $c$  denotes parameter held in common across all rows/columns.

| name               | type     | variance matrix<br>(for set of $n$ effects)               | description                               |
|--------------------|----------|---|---|
| <code>id()</code>  | unscaled | $\mathbf{U} = \mathbf{I}$                                 | IID with variance 1.                      |
| <code>idv()</code> | scaled   | $\mathbf{V} = \sigma_c^2 \mathbf{I}$                      | IID with common variance = default model. |
| <code>idh()</code> | scaled   | $\mathbf{V} = \text{diag}\{\sigma_1^2 \dots \sigma_n^2\}$ | independent with separate variances.      |
| <code>ar1()</code> | unscaled | $C_{ij} = \rho_c^{ i-j }$                                 | auto-regressive structure of order 1.     |

## 7.2 Process to define a consolidated model term

| name                | type     | variance matrix<br>(for set of $n$ effects)      | description   |
|---------------------|----------|--|---|
| <code>arlv()</code> | scaled   | $V_{ij} = \sigma_c^2 \rho_c^{ i-j }$             | auto-regressive structure of order 1.   |
| <code>arlh()</code> | scaled   | $V_{ij} = \sigma_i \sigma_j \rho_c^{ i-j }$      | auto-regressive structure of order 1.   |
| <code>corg()</code> | unscaled | $C_{ij} = \rho_{ij}$                             | unstructured correlation matrix.  |
| <code>diag()</code> | scaled   | $V = \text{diag}\{\sigma_1^2 \dots \sigma_n^2\}$ | independent with separate variances (same as <code>idh()</code> ).  |
| <code>grmk()</code> | unscaled | $S$ specified                                    | applies a known scaled variance matrix (typically a <b>GRM</b> matrix); the number of rows in the matrix must 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. When multiple <b>GRM</b> matrices are defined, use $k$ (default 1) to identify which one to use. |
| <code>givk()</code> | unscaled | $S$ specified                                    | applies a known inverse of a scaled variance matrix (typically the <b>GRM inverse</b> matrix). When multiple matrices are defined, use $k$ (default 1) to identify which one to use.  |
| <code>nrm()</code>  | unscaled | $S$ specified                                    | applies a generated pedigree-based numerator relationship matrix <b>NRM</b> derived from the functions argument associated pedigree file.   |
| <code>ide()</code>  | scaled   | $V = \sigma_c^2 I$                               | applies <b>IID</b> with common variance rather than using any <b>NRM/GRM</b> with covariance matrix associated with the argument of the function.   |
| <code>rrk()</code>  | scaled   | $V = \Gamma \Gamma^T + \Psi$                     | factor analytic model of order $k$ with $\Gamma$ of size $n \times k$ and $\Psi$ is an $n \times n$ diagonal matrix with all zero elements.   |
| <code>us()</code>   | scaled   | $V_{ij} = \sigma_{ij}$                           | general unstructured, symmetric positive definite covariance matrix.  |
| <code>xfak()</code> | scaled   | $V = \Gamma \Gamma^T + \Psi$                     | factor analytic model of order $k$ with $\Gamma$ of size $n \times k$ and $\Psi$ is an $n \times n$ diagonal matrix.  |

The variance structure associated with each component in Table 7.2 has a 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. The simplest form of a consolidated model term is a single model term with a variance model function applied, e.g. `idv(repl)` in Table 7.2, and the next simplest is a compound model term with a variance model function applied, e.g. `idv(A.B)` in Table 7.2.

Table 7.2: Building consolidated model terms in ASReml

|   | linear model term<br>(type of term)  | component(s)        | variance<br>structure name | variance model<br>function name | covariance<br>component   | type | consolidated model term           |
|---|--------------------------------------|---------------------|----------------------------|---------------------------------|---------------------------|------|-----------------------------------|
| 1 | <code>repl</code><br><i>single</i>   | <code>repl</code>   | IDV                        | <code>idv()</code>              | <code>idv(repl)</code>    | S    | <code>idv(repl)</code>            |
| 2 | <code>fac(x)</code><br><i>single</i> | <code>fac(x)</code> | EXPV                       | <code>expv()</code>             | <code>expv(fac(x))</code> | S    | <code>expv(fac(x))</code>         |
| 3 | <code>A.B</code><br><i>compound</i>  | <code>A.B</code>    | IDV                        | <code>idv()</code>              | <code>idv(A.B)</code>     | S    | <code>idv(A.B)</code>             |
| 4 | <code>column.row</code>              | <code>column</code> | IDV                        | <code>idv()</code>              | <code>idv(column)</code>  | S    | <code>idv(column).ar1(row)</code> |

## 7.2 Process to define a consolidated model term

|          | linear model term<br>( <i>type of term</i> ) | component(s)    | variance<br>structure name | variance model<br>function name | covariance<br>component   | type   | consolidated model term |
|----------|--|-----------------|----------------------------|---------------------------------|---------------------------|--------|-------------------------|
|          | <i>compound</i>                              | row             | <b>ARI</b>                 | arl()                           | arl(row)                  | U      |                         |
| <b>5</b> | site.variety<br><i>compound</i>              | site<br>variety | <b>DIAG<br/>ID</b>         | diag()<br>id()                  | diag(site)<br>id(variety) | S<br>U | diag(site).id(variety)  |
| <b>6</b> | Trait.animal<br><i>compound</i>              | Trait<br>animal | <b>US<br/>NRM</b>          | us()<br>nrm()                   | us(Trait)<br>nrm(animal)  | S<br>U | us(Trait).nrm(animal)   |

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

- Scaled variance model functions can be applied to single model terms (see example **1** in Table 7.2), single model terms with a constructed linear model function (example **2**), compound model terms use one component having a scaled model function and the other components having unscaled model functions (examples **4** to **6**). For convenience the type of model function, scaled (*V*) or unscaled (*U*) is indicated in the Table 7.2 (*column 6*).
- Scaled variance model functions can also be applied to compound model terms (example **3**).
- Variance model functions *cannot* be applied to expandable model terms, for example
  - $A*B$  which expands to  $A \ B \ A.B$
  - $A/B$  which expands to  $A \ A.B$
  - $at(A, i, j).B$  which expands to  $at(A, i).B \ at(A, j).B$
- A scaled variance function can be specified for one, but only one, component in a compound model term. Unscaled variance functions must be defined for the remaining terms. This is due to the identifiability issues that occur when compound variance structures are specified. This is explained in NIN example **3a**, see Section 7.5. The scaled 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.
- If a compound model terms only includes unscaled variance functions, then **ASReml** will insert a variance/scale parameter. For example, for single mode terms using the functions `grm()`, or `nrm()` both will have a variance parameter added.
- For consolidated model terms involving more than two functions, it is important to place first `grm()`, `nrm()`, or `xfm()`.

### 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 **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/(un)consolidated model(s))`

The  $m$  individual model terms generate the design matrices  $\mathbf{Z}_i$  and effect vectors  $\mathbf{u}_i$  of size  $b_i$  ( $i = 1, \dots, m$ ) and the  $v$  variance structure terms generate variance structures  $\mathbf{G}_j$  of size  $b_j^*$  ( $j = 1, \dots, v$ ). The function `str()` generates a combined model design matrix  $\mathbf{Z}_c = [\mathbf{Z}_1 \dots \mathbf{Z}_m]$  and a combined effects vector  $\mathbf{u}_c^\top = [\mathbf{u}_1^\top \dots \mathbf{u}_m^\top]$  of size  $b_c = \sum_{i=1}^m b_i$  and the variance structure for  $\mathbf{u}_c$  is  $\mathbf{G}_c = \bigoplus_{j=1}^v \mathbf{G}_j$  for  $\mathbf{u}_c$  and  $\mathbf{G}_c$  to be conformable  $\sum_{j=1}^v b_j^* = b_c$ . If  $v = 1$  then there is one variance structure associated with the combined set of effects and if  $v > 1$  we can partition  $\mathbf{u}_c$  and  $\mathbf{G}_c$  with  $\mathbf{u}_c^\top = [\mathbf{u}_1^\top \dots \mathbf{u}_v^\top]$  and  $\mathbf{G}_c = [\mathbf{G}_1^* \dots \mathbf{G}_v^*]$  and the effect vectors are independent of each other and the effects  $\mathbf{u}_j^*$  have variance structure  $\mathbf{G}_j^*$ .

### 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.

Note that in the first term (`Animal`) we have the intercepts, the second term (`age.Animal`) has the slopes, and together they define a 2 by 10 matrix. In the third term, (`us(2):id(10)`), the `us(2)` structure defines the two variances and the covariance in the (intercept, slope) dimension, and the `id(10)` defines an IID structure for the second dimension. In our example, this random model generates a combined set of random effects from the individual animal intercepts,  $\mathbf{u}_I = (u_{I1} \dots u_{I10})^\top$  and animal slopes,  $\mathbf{u}_S = (u_{S1} \dots u_{S10})^\top$ , as  $\mathbf{u}_{IS} = (\mathbf{u}_I^\top \dots \mathbf{u}_S^\top)^\top$ . The consolidated term then has variance structure of the form

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

Here, the set of animal intercepts has a common variance ( $\sigma_{II}$ ), and the set of animal slopes has a (different) common variance ( $\sigma_{SS}$ ). Intercepts and/or slopes from two different animals are independent, but the intercept and slope from any given animal have covariance  $\sigma_{IS}$  (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, e.g. `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))
```

if `Animal`, `Sire` and `Dam` are coded conformably, *i.e.* `Sire` and `Dam` defined to have the same number of levels as `Animal` and the levels of `Sire` and `Dam` are a subset of the levels of `Animal` (perhaps by using the same labels).

Note that the above model is associating the `Animal` term to a **NRM** matrix and also using directly this term instead of `nrm(10)`. 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)).nrm(Animal)
```

or

```
us(leg(age)).nrm(Animal)
```

except the intercept/slope design matrix is different in this last case.

More generally one can use consolidated model terms with variable names to specify variance structures with the one proviso that in order to simplify processing of the 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, *e.g.* `us(2)` in the original example. We note that in one sense we are using unconsolidated model terms or variance structure terms as with an integer in the function as only a variance structure and not a design matrix can be specified.

### 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))
```

A rather artificial example of using  $\nu$  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 ( $AR1 \times AR1$ ) 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

## 7.3 Applying variance structures to the residual error term

---

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  $(AR1 \times AR1)$  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 reorder the data internally when dimensions are used but instead assumes that the specified variance structure matches the order of the data as presented in the data file.

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 Special properties and rules in defining the residual error term

There are certain properties and associated rules for this term 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 5.5) 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,  $R_v = R_{v_1} = \sigma^2 I_n$ , see Example 2.2, defined either by default (*i.e.* 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 2.1.4 described partitioning the data observations into data *sections* to which separate variance structures are applied. There are three data *sections* in the fourth example in Section 7.3. 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** (*i.e.* the data file does not need to be ordered in any particular way) and the variance structures for sections can then be specified using the `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  $\text{AR1} \times \text{AR1}$  variance structure, we could write

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

Alternatively, an  $\text{AR1} \times \text{AR1}$  variance structure for sites 1 and 3, but an  $\text{IDV} \times \text{AR1}$  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).ar1v(column).id(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  $\text{AR1}$  matrices.

**Important** 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.

The use of the function `sat()` allows the concise specification of residual models. In some cases, for example

```
residual sat(section).ar1v(row)
```



there might only be one level of the autoregressive argument in some sections and so the autoregressive parameter cannot be estimated. In these sections **ASReml** now fixes the autoregressive parameter but does not use it.

### 7.3.3 Using !VPGROUP to constrain variance components

There is an additional qualifier `!VPGROUP f1 [f2] [f3]` that facilitates the constraints of variance parameters in `sat()` terms to common values.

For the residual model

```
sat(Exp).ar1(Row).ar1v(Col)
```

when variance parameters associated with experiments can be classified into *groups* the user may wish to constrain the variance parameters to be the same for *experiments* in the same *group*. `!VPGROUP` may have up to 3 arguments, which are applied in order. For example,

```
sat(Exp !VPGROUP Group1 Group2 Group3).ar1(Row).ar1v(Col)
```

where `Groupi` ( $i = 1, 2, 3$ ) are variables that group the data associated with `Exp`. This example has 3 spatial parameters for each experiment in the order: row correlation, column correlation and variance; **ASReml** will then constrain the  $i$ -th parameter according to `Groupi`. For example,

```
sat(Exp !VPGROUP mu Group Exp).ar1(Row).ar1v(Col)
```

allows a common row autocorrelation across all sections, a field specific column autocorrelation and a different variance for each section. If the same grouping is applied to all parameters, the specification can be reduced to

```
sat(Exp !VPGROUP Group1).ar1(Row).ar1v(Col)
```

This qualifier can also be used with `sat()` and `grm()`, among others.

## 7.4 Identifiability

Once all components of a model term have a variance function specified, **ASReml** attempts to check whether the variance parameters are identifiable. If no variance parameter is found as in `id(A).ar1(B)` one is added; either fitting `id(A).ar1v(B)` or `idv(A).ar1(B)`. Some users might find easier to directly specifying terms as variance terms as above. If more than one variance model function in the consolidated model term, such as `idv(A).ar1v(B)`, **ASReml** will give a warning if this is detected. In this case, the **AI** matrix will be singular and **ASReml** will abort (unless `!AISING` is set or one of the variances is fixed. Here, as the parameters will not all be identifiable, 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

$$\mathbf{e} \sim N(\mathbf{0}, \sigma_e^2 \mathbf{I}_{224})$$

The model therefore involves just one **R** structure (IDV) and no **G** structure as `variety` and `repl` are fitted as fixed effects. 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

$$\mathbf{u}_r \sim N(\mathbf{0}, \sigma_r^2 \mathbf{I}_4)$$

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 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

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 and so we use the augmented version of the data file (see page 24) which has records for all plots in the complete grid. Alternative can fill out the data by using `!ROWFACTOR` and `!COLUMNFACTOR` (see Table 5.5). 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

$$\mathbf{e} \sim N(\mathbf{0}, \sigma_{e_c}^2 \mathbf{I}_{11} \otimes \boldsymbol{\Sigma}_r(\rho_r))$$

that is, a two-dimensional first order separable autoregressive spatial structure for error but with spatial correlation in the row direction only (IDV×ARI).

`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

$$\text{var}(\mathbf{e}) = \sigma_{e_c}^2 \mathbf{I}_{11} \otimes \boldsymbol{\Sigma}_r(\rho_r)$$

### Important

- The same residual variance structure could be achieved by specifying

`id(column).ar1v(row)`

which mirrors the alternate but equivalent algebraic form

$$\text{var}(\mathbf{e}) = \mathbf{I}_{11} \otimes \sigma_{e_r}^2 \boldsymbol{\Sigma}_r(\rho_r)$$

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

$$\text{var}(\mathbf{e}) = \sigma_e^2 \mathbf{I}_{11} \otimes \boldsymbol{\Sigma}_r(\rho_r)$$

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.

- It is technically an error if the consolidated model term

`idv(column).ar1v(row)`

was specified. This would correspond to

```
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)
```

$$\text{var}(\mathbf{e}) = \sigma_e^2 \mathbf{I}_{11} \otimes \sigma_{e_r}^2 \boldsymbol{\Sigma}_r(\rho_r)$$

and  $\sigma_e^2$  and  $\sigma_{e_r}^2$  are unidentifiable in this case, that is, it is not possible to estimate them separately.

- In a univariate single section analysis like this, **ASReml** by default uses the gamma parameterization for estimation, see Section 7.6, and no error is flagged with only one variance reported. The user can force **ASReml** to use the sigma parameterization by placing `!SIGMAP` (Section 7.6) before the model line or on the model line before `~`.

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

Note that with `!SIGMAP` or with multiple sections, the attempt to fit two variance parameters is reported and the job is aborted because of the consequent singularity in the **AI** matrix.

### 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

$$\text{var}(\mathbf{e}) = \sigma_{e_c}^2 \boldsymbol{\Sigma}_c(\rho_c) \otimes \boldsymbol{\Sigma}_r(\rho_r)$$

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  $\sigma_{e_c}^2 \boldsymbol{\Sigma}_c(\rho_c)$  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

- The same residual variance structure could be achieved by specifying

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

which mirrors the alternate but equivalent algebraic form

$$\text{var}(\mathbf{e}) = \boldsymbol{\Sigma}_c(\rho_c) \otimes \sigma_{e_r}^2 \boldsymbol{\Sigma}_r(\rho_r)$$

- 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 under the sigma parameterization if the consolidated model term

```
ar1v(column).ar1v(row)
```

was specified as this would correspond to

$$\text{var}(\mathbf{e}) = \sigma_{e_c}^2 \mathbf{\Sigma}_c(\rho_c) \otimes \sigma_{e_r}^2 \mathbf{\Sigma}_r(\rho_r)$$

and  $\sigma_{e_c}^2$  and  $\sigma_{e_r}^2$  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

$$\text{var}(\mathbf{u}_r) = \sigma_r^2 \mathbf{I}_4, \text{var}(\mathbf{u}_\eta) = \sigma_\eta^2 \mathbf{I}_{242}$$

and

$$\text{var}(\mathbf{e}) = \sigma_{e_c}^2 \mathbf{\Sigma}_c(\rho_c) \otimes \mathbf{\Sigma}_r(\rho_r)$$

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  $\sigma_\eta^2 \mathbf{I}_{242}$ . 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

`arlv(column).arl(row)`

This order is reversed in **4**. Sometimes this model can be hard to fit, particularly when there is little spatial correlation.

### 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

$$\begin{aligned} \text{var}(\mathbf{u}_r) &= \sigma_r^2 \mathbf{I}_4, \\ \text{var}(\mathbf{u}_{cr}) &= \sigma_{c_r}^2 \mathbf{\Sigma}_c(\rho_c) \otimes \mathbf{\Sigma}_r(\rho_r) \text{ and} \\ \text{var}(\mathbf{e}) &= \sigma_e^2 \mathbf{I}_{242} \end{aligned}$$

```
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 arlv(column).arl(row)
```

```
NIN Alliance Trial 1989
  variety !A
  id
  :
  row 22
  column 11
  nin89aug.asd !SKIP 1
  yield ~ mu variety !r idv(repl),
  arlv(column).arl(row) !f mv
  residual idv(units)
```

#### Important

- The same **G** structure could be achieved by specifying

`arl(column).arlv(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 gammas and sigmas are reported in the variance components table. 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 2.4, the variance matrix of  $\mathbf{y}$  is

$$\text{var}(\mathbf{y}) = \mathbf{ZG}(\boldsymbol{\sigma}_g)\mathbf{Z}^\top + \mathbf{R}_v(\boldsymbol{\sigma}_r),$$

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

$$\mathbf{R}_v(\boldsymbol{\sigma}_r) = \sigma_e^2 \mathbf{R}_c(\gamma_r)$$

the variance matrix of  $\mathbf{y}$  can be written alternatively as

$$\text{var}(\mathbf{y}) = \sigma_e^2 (\mathbf{ZG}(\gamma_g)\mathbf{Z}^\top + \mathbf{R}_c(\gamma_r))$$

see (2.8). This is referred to as the *gamma parameterization* and the variance structure parameters in  $\gamma_g$  and  $\gamma_r$  are referred to as *gammas*, see Section 2.1.6.

### 7.6.1 Which variance parameterization is used during estimation?

By default, **ASReml** uses sigma parameterization unless there is only one residual variance parameter, in which case the default is the gamma parameterization. There is only one variance parameter if the analysis is univariate and there is only 1 section of data. Forcing the sigma parameterization is discussed in the next 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 (*i.e.* scaled identity).

### 7.6.2 Switching from the gamma to the sigma parameterization

For a univariate, unweighted single section analyses, **ASReml** uses the gamma parameterization unless `!SIGMAPAR` is specific. For example

## 7.6 Sigma versus gamma parameterization

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

will use the sigma parameterization, and

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

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

Fitting a model with the gamma parameterization is more robust to handle data with variance outside the range 0.01 to 1000. **Error! Reference source not found.** gives the variance model specification for each of the six NIN examples (column 3), the individual terms in  $\mathbf{G}(\sigma_g)$  and  $\mathbf{R}_v(\sigma_r)$  under the sigma parameterization (column 4), the sigmas that are estimated under this parameterization (column 5), the individual terms in  $\mathbf{G}(\gamma_g)$  and  $\mathbf{R}_v(\gamma_r)$  under the gamma parameterization (column 6) and the gammas that are estimated under this parameterization (column 7).

Table 7.3:  $\mathbf{G}$  structure for the random terms (magenta) and  $\mathbf{R}$  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

| no | definition  | variance model specification   | sigma parameterization  |   | gamma parameterization   |   |
|----|---|--|---|---|--|---|
|    |   |  | $\mathbf{G}(\sigma_g)$<br>$\mathbf{R}_v(\sigma_r)$  | sigma(s)  | $\mathbf{G}(\gamma_g)$<br>$\mathbf{R}_c(\gamma_r)$   | gamma(s)  |
| 1  | RCB analysis: blocks fixed  | <code>residual idv(units)</code>   | $\sigma_e^2 \mathbf{I}_{224}$   | $\sigma_e^2$  | $\mathbf{I}_{224}$   | none  |
| 2  | RCB analysis blocks random  | <code>!r idv(repl)</code><br><code>residual idv(units)</code>  | $\sigma_r^2 \mathbf{I}_4$<br>$\sigma_e^2 \mathbf{I}_{224}$  | $\sigma_r^2$<br>$\sigma_e^2$  | $\gamma_r \mathbf{I}_4$<br>$\mathbf{I}_{224}$  | $\gamma_r$                                      |
| 3a | Two-dimensional spatial model correlation in one direction                      | <code>!r idv(repl)</code><br><code>residual idv(column).ar1(row)</code>                              | $\sigma_r^2 \mathbf{I}_4$<br>$\sigma_{e_c}^2 \mathbf{I}_{11} \otimes \Sigma_c(\rho_r)$                                      | $\sigma_r^2$<br>$\sigma_{e_c}^2, \rho_r$                            | $\gamma_r \mathbf{I}_4$<br>$\mathbf{I}_{11} \otimes \Sigma_r(\rho_r)$                                      | $\gamma_r$<br>$\rho_r$                          |
| 3b | Two-dimensional separable autoregressive spatial model                          | <code>!r idv(repl)</code><br><code>residual ar1v(column).ar1(row)</code>                             | $\sigma_r^2 \mathbf{I}_4$<br>$\sigma_{e_c}^2 \Sigma_c(\rho_c) \otimes \Sigma_r(\rho_r)$                                     | $\sigma_r^2$<br>$\sigma_{e_c}^2, \rho_r, \rho_c$                    | $\gamma_r \mathbf{I}_4$<br>$\Sigma_c(\rho_c) \otimes \Sigma_r(\rho_r)$                                     | $\gamma_r$<br>$\rho_r, \rho_c$                  |
| 3c | Two-dimensional separable autoregressive spatial model with measurement error   | <code>!r idv(repl),</code><br><code>idv(units)</code><br><code>residual ar1v(column).ar1(row)</code> | $\sigma_r^2 \mathbf{I}_4$<br>$\sigma_\eta^2 \mathbf{I}_{224}$<br>$\sigma_{e_c}^2 \Sigma_c(\rho_c) \otimes \Sigma_r(\rho_r)$ | $\sigma_r^2$<br>$\sigma_\eta^2$<br>$\sigma_{e_c}^2, \rho_r, \rho_c$ | $\gamma_r \mathbf{I}_4$<br>$\gamma_\eta \mathbf{I}_{234}$<br>$\Sigma_c(\rho_c) \otimes \Sigma_r(\rho_r)$   | $\gamma_r$<br>$\gamma_\eta$<br>$\rho_r, \rho_c$ |
| 4  | Two-dimensional separable autoregressive spatial model defined as a G structure | <code>!r idv(repl),</code><br><code>ar1v(column).ar1(row)</code><br><code>residual idv(units)</code> | $\sigma_r^2 \mathbf{I}_4$<br>$\sigma_{cr_c}^2 \Sigma_c(\rho_c) \otimes \Sigma_r(\rho_r)$<br>$\sigma_e^2 \mathbf{I}_{224}$   | $\sigma_r^2$<br>$\sigma_{cr_c}^2, \rho_r, \rho_c$<br>$\sigma_e^2$   | $\gamma_r \mathbf{I}_4$<br>$\gamma_{cr_c} \Sigma_c(\rho_c) \otimes \Sigma_r(\rho_r)$<br>$\mathbf{I}_{224}$ | $\gamma_r$<br>$\gamma_{cr_c}, \rho_r, \rho_c$   |

## 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 7.10).
- `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.

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.

Normally the first argument in a covariance component function is the name of a variable/model term which defines the design matrix for the effects to which the variance structure is applied. Sometimes, there is no such variable in which case the first argument is the size/dimension,  $\omega$  say, of the structure. Then a variance structure of a particular dimension,  $\omega$  say, can be specified directly as

`vmfname( $\omega$  qualifiers)`

For example

`idv(3)`

defines the IDV variance structure of dimension 3, that is,  $\sigma^2 \mathbf{I}_3$ , and

`idv(3 !INIT 1.1)`

specifies an initial value of 1.1 for the associated variance component under the sigma



## 7.7 Variance model function qualifiers

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  $\sigma^2$  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.

**Table 7.4: Variance model function qualifiers available in ASReml**

| qualifier                  | description   |
|----------------------------|---|
| <code>!=s</code>           | <i>s</i> is a list of codes that link parameters sharing a common value; details in Section 7.7.1.  |
| <code>!COORD v</code>      | 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.<br><br><code>exp(Trait !COORD 1 2.5 3.5 5 8)</code><br><br>see Section 7.7.2.  |
| <code>!Fi</code>           | is used with the <code>own()</code> variance model function, see Section 7.7.3. The argument <i>i</i> is passed to your <i>own</i> program.   |
| <code>!Gs</code>           | <i>s</i> is a list of codes F, Z, H, P or U, one for each parameter, specifying whether the parameter is to be <i>Fixed</i> at its initial value, held at <i>Zero</i> (if legal), <i>Held</i> for a few iterations, kept in the <i>Parameter</i> space or is <i>Unrestricted</i> , see Section 7.7.4. |
| <code>!INIT v</code>       | <i>v</i> is the list of initial values for the variance structure parameters. If initial values can be obtained from the <code>.msv</code> , <code>.rsv</code> or <code>.tsv</code> file, they override these values, see Section 7.7.5.  |
| <code>!SUBSECTION f</code> | <i>f</i> is a factor in the data that breaks the section into independent subsections, with subsections having common variance parameters, now deprecated and replaced by <code>sat(!VPGROUP)</code> , see Section 7.3.3).  |
| <code>!Ts</code>           | is used with the <code>own()</code> variance model function to set the parameter types, see Sections 7.7.6 and 7.7.3.   |
| <code>!USE t</code>        | <i>t</i> 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 <i>t</i> , see Section 0 for an example.   |

### 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

- 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

## 7.7 Variance model function qualifiers

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 (Section 7.8.1) is required for these cases.

Table 7.5: Examples of constraining variance parameters in ASReml

| ASReml code   | action  |
|---|---|
| <code>!=ABACBADCB</code>  | 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 x 4 matrix would make it banded, that is<br><pre> A B A C B A D C B A </pre>  |
| <code>us(site !GP !=0A0AA0,<br/>!INIT .3 .1 .4 .1 .1 .3)</code>   | 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.  |
| <code>fa2(site !G4PZ3P4P !=0000000V VVV,<br/>!INIT 4*.9 0 3*.1 4*.2).gen</code>                                       | a factor analytic model of order 2 for 4 sites, with equal variance across sites, is specified using this code. For the <code>fa</code> variance model functions, ASReml orders the parameters as <i>the loadings followed by the specific variances</i> . 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.                                  |
| <code>xfa2(site !=VVVV00000000,<br/>#contracted form #!=*4V8.<br/>!4P4PZ3P,<br/>!INIT 4*0.2 4*1.2 0 3*0.3).gen</code> | code for a factor analytic model of order 2 for 4 sites, in which the specific variances are all equal. For the <code>xfa</code> variance model functions, ASReml orders the parameters as <i>the specific variances followed by the loadings</i> (note that this is different to the ordering for the <code>fa</code> 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 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 !F *i*

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 *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 *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
row 10
col 15
variety 25
yield
barley.asd !SKIP 1 !OWN MYOWNGDG.EXE !MAX 30
y ~ variety
residual ar1(10 0.5).own2(15 !INIT 0.2 0.1 !TRR !F1)
```

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

```
15      2      1
0.6022550D+00 0.1167998D+00
```

```
This file was written by ASReml for reading by your program MYOWNGDG.EXE
ASReml writes this file, runs MYOWNGDG.EXE and then reads barleyOWNB.gdg
which it presumes you have written in 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 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 (6D14.7) format.
Follow this with      3 matrices written in (6D14.7) 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 7.7.6, 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.

`!Ts` 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 7.7.6.

### 7.7.4 Parameter space constraints `!Gs`

Each parameter has an associated *constraint* code which may be expressed explicitly with the qualifier `!Gs`, where *s* is the code. Table 7.6 shows the possible constraint codes.

Table 7.6: Variance parameter space constraint options

| 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 <code>!GP</code> condition, repeated attempts to make a variance negative are detected and the value is then <i>fixed</i> 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 $\pm 1$ . Negative variance components may lead to problems; the mixed model coefficient matrix may become non-positive definite. In this case the sequence of <b>REML</b> log-likelihoods may be erratic and you may need to experiment with starting values.   |
| F    | fixed              | F fixes the parameter at its starting value.  |
| H    | fixed/in the space | H holds the parameter for the first <i>h</i> iterations and then the code changed to P so that the parameter can be estimated. H has a default value of 3; the value can be set with the <code>!FREEGH</code> data line qualifier. This was introduced to facilitate calculation in GLM models when is sometimes the need in the first few iterations to focus on constructing the working variables.   |
| 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 `!GXXXX` 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 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), or by supplying an `.rsv` file using `!CONTINUE` (see Section 7.9.2). Note that initial values taken from a `.tsv`/`.msv`/`.rsv` file overwrite those supplied by `!INIT`.

#### Important

- 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; expressed relative to the residual variance (*i.e.* ratios). For example, a genetic variance of 30 expressed relative to a variance of 240 has a gamma value of 0.125.

### 7.7.6 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*.

Table 7.7: Variance components parameter types

| 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    |                      |

### 7.7.7 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 `xfal(dtrial)` and both must have the same parameters.

That is, instead of fitting

```
xfal(dTrial).grm3(entry)
```

we fit

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

requiring both `xfal` 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
xfal(dTrial $QP $QE $QI).grm1(family),
xfal(dTrial $QP $QE $QI).grm2(entry)
```

However, for a larger term, the number of parameters required may exceed the available letters in the alphabet. One option is to use the VCC machinery (Section 7.8.1) as shown below.

```
xfal(dTrial $QP $QI).grm1(family),
xfal(dTrial $QP $QI).grm2(entry)
...
VCC xfal(dTrial).grm1(family) 1 9 !BLOCKSIZE 8
```

A better option is to use just one structure twice with the !USE qualifier, to equate the parameters sets when writing the terms in the model line. The following code associates

```
xfal(dTrial) in xfal(dTrial).giv2(entry)
```

with

```
xfal(dTrial) in xfal(dTrial).giv1(family)
```

that is, both terms point to the one structure definition, i.e. point to the same variance parameters'

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

Table 7.5 gives examples of constraining variance parameters in ASReml.

## 7.8 Setting relationships among variance structure parameters

The aim of this section is to explain how variance parameters may be linked to have the same value or a proportional value, especially across model terms. Section 7.7.1 discussed the simplest case where a few parameters in one variance structure; for example, covariances in an otherwise unstructured variance matrix, are to have a common value. More general cases are discussed in the next two subsections. Note that all variance parameters are held in a single vector. The parameter position numbers and their associated fully specified names are listed in the `.tsv` file.

### 7.8.1 Simple relationships among variance structure parameters

It is possible to define simple equality relationships among variance parameters in a particular variance structure using the `!=s` qualifier, see Section 7.7.1 and Table 7.4. Formal linear relationships between variance structure parameters can be defined by placing the `VCC` or `VCM` directives after the residual line. Unlike the case of parameter equality, all parameters can be accessed and the linear relationship is not limited to equality. Since the parameter positions (which are given in the `.tsv` file) are not easily anticipated, the `VCC` statement begins by identifying the first model term containing a parameter to be linked to other parameters. It may be necessary to use the `str()` model function to hold other model terms in their relative position since the particular parameters to be linked are specified relative to the first parameter in the ‘first model term’.

The syntax is

```
VCC term !LINK relative position [!SCALE scales] [!BLOCKSIZE s]
```

or

```
VCC termA [-] termB
```

- The `VCC` statement must occur after the ‘residual’ statement. Multiple `VCC` statements are permitted but the linked parameter sets must be independent (otherwise see `VCM` in Section 7.8.2).
- `term` is the model term containing the first parameter to be linked.
- `termA` is a model term with a single variance parameter.
- `termB` is another model term with a single variance parameter which is to be made equal to that for `termA`, or the same but with opposite sign.
- `!LINK relative positions` is a list of parameter positions where 1 is the first parameter in `term`. For equally spaced positions, use ‘:’ to represent positions between the second and the last; the increment is taken from the initial interval.

## 7.8 Setting relationships among variance structure parameters

- `!SCALE` *scales* used when the relationship required among linked parameters is not equality. Hence, supply the relative size coefficient for each member of the `LINK` list.

For example

```
VCC diag(Env).grm3(FMF) !LINK 1:10 !SCALE 1 2 2 1 2 2 1 2 3 1
```

requires `Env` classes 1, 4, 7 and 10 to have the same variance, classes 2, 3, 5, 6 and 8 to have a variance twice that for class 1, and class 9 to have a variance 3 times that for class 1. If `s` is the vector of scale values supplied, `s/s[1]` is the vector of scale values actually used. Scale values may be integrated into the `!LINK` list, changing the example to read

```
VCC diag(Env).grm3(FMF) !LINK 1 2*2 3*2 4 5*2 6*2 7 8*2 9*3 10
```

- `!BLOCKSIZE` `b` is provided for the situation where there are `g` groups of `b` parameters and we wish to constrain the first parameter of each group numbered (say,  $s_1, s_2, \dots, s_g$ ) and then constrain the second parameter, the third, and so on. Hence, `!BLOCKSIZE` allows to constrain the first set of parameters and then **ASReml** will generate the constraints for the other `b-1` sets.

For example

```
VCC rr(Exp) !LINK 1 4 7 10 !BLOCKSIZE 3
```

generates

```
VCC rr(Exp) !LINK 1 4 7 10
VCC rr(Exp) !LINK 2 5 8 11
VCC rr(Exp) !LINK 3 5 9 12
```

The initial motivation for `!BLOCKSIZE` was constraining parameters generated by `sat()` terms on the residual line but the numbering can be counter-intuitive. This can now be achieved much easily using the `!VPGROUP` qualifier within `sat()`.

Table 7.8: VCC directive examples

| VCC statement  | action  |
|--|---|
| <pre>VCC idv(units) !LINK 1 2 !SCALE 1 -1 VCC idv(units) !LINK 1 -2 VCC idv(units) -uni(Check)</pre> | <p>The parameter following units is to have the same magnitude but opposite sign. The four lines are equivalent for a model line</p> <pre>y ~ mu Cov Ch mv !r, idv(units !INIT 1), uni(Check !INIT -1) residual arlv(Col).id(Row)</pre> |
| <pre>VCC idv(units) -uni(Check)</pre>  |   |
| <pre>VCC us(Env).idv(blocks) !LINK 2 4 5 7 8 9</pre>   | <p>For a <math>(4 \times 4)</math> US matrix <code>us(Env)</code>, the covariances are made equal.</p>  |

The VCC syntax only caters for the case where a set of parameters are all (scaled) versions of the same parameter. The information is combined from all parameters to estimate the common value and sets must therefore be independent. The VCM qualifier in the next section allows for parameters to be linked via a matrix.

The former (**ASReml 3**) VCC qualifiers may be required when the user cannot see how to specify the required constraints using the VCC *term* `!LINK` *relative position* syntax above. They are



## 7.8 Setting relationships among variance structure parameters

- *linklist* replaces “*term* !LINK *relative position* [!SCALE *scales*]” by explicitly giving the required list of parameter numbers as identified from the .tsv file. Any scale values are given after ‘\*’ attached to the *parameter number*. Indeed this alternative specification of SCALE can be used either way.
- [!OFFSET *o*] is deprecated as not needed when *term* is given. It was used when a change of model moved the *linklist* numbers up or down by *o* as an alternative to retyping the list.

When variance parameters are linked across model terms, the order of model terms must be known and is found in the .tsv file. The order of model terms may not be the order in which they were specified in the model line unless formally associated using the !{ and !} syntax. Parameters linked by VCC statements are indicated in the variance component table by “C” in the “C” column and the parameter number they are linked to.

Some examples using the legacy and new instructions are given in the following table

Table 7.9: Legacy and new VCC directive examples

| VCC statement  | action   |
|--|--|
| VCC 5 6  | Parameters 5 and 6 are the same.   |
| VCC 5 6 !SCALE 1 -1<br>VCC 5 6*-1<br>VCC 5 -6  | Parameters 5 and 6 have the same magnitude but opposite sign. The four lines are equivalent for a model line |
| VCC idv(units) -uni(check)   | y ~ mu Cov Ch mv !r<br>idv(units !INIT 1),<br>uni(Check !INIT -1)<br>residual arlv(Col):id(Row)              |
| VCC at(expt,18).arlv(prange).arl(prow)<br>!LINK 1 4:13 !BLOCKSIZE 3<br>VCC 62 65:74 !BLOCKSIZE 3 | Two equivalent VCC statements using new and legacy syntax.   |

### 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.

## 7.8 Setting relationships among variance structure parameters

---

- *new* is the number of new parameters.
- *filename* is the name of the file containing the design matrix.

For example, the Wolfinger rats example involves modelling a  $5 \times 5$  symmetric residual matrix.

```
Wolfinger Rat data
treat !A
wt0 wt1 wt2 wt3 wt4
subject * !=V0
wolfrat.dat !SKIP 1
wt0 wt1 wt2 wt3 wt4 ~ Trait treat Trait.treat
residual units.us(Trait)
# It 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 Huynh-Feldt variance structure to this data. This structure is of the form

$$\sigma_{ii} = \sigma_{ni}$$
$$\sigma_{ij} = \frac{1}{2}(\sigma_{ni} + \sigma_{nj} - \sigma_{no}), \quad j < i \leq p, o = p + 1$$

In the rats example  $p = 5$ , the relationship between the original and new parameters is  $\sigma = M\sigma_n$  where  $\sigma$  and  $\sigma_n$  are  $p(p + 1)/2 \times 1 = 15 \times 1$  and  $(p + 1) \times 1 = 6 \times 1$  vectors respectively, and  $M$  is a  $p(p + 1)/2 \times (p + 1) = 15 \times 6$  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 replace the model specification lines by

```
# Supply start values because raw SSP generates bad initial values
# for HuynhFeldt structure because it does not fit well
!ASSIGN HFvcm !GU !INIT 45 20 45 20 20 45 20 20 45 20 20 20 45
wt0 wt1 wt2 wt3 wt4 ~ Trait treat Trait.treat
residual units.us(Trait $HFvcm)
VCM 5 19 6 HuynhFeldt.vcm # Parameters 5 to 19 explained in terms of 6 parms.
```

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 7.9.1.

### 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
wolfstat.dat !SKIP 1 !ASUV !MAXITER 0
wt0 wt1 wt2 wt3 wt4 ~ Trait treat Trait.treat
residual units.us(Trait)
```

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 !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, F,      1.0000000,      ,      4,      1
  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
#
# 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.
```

## 7.9 Ways to present initial values to ASReml

Parameter constraints and initial values can be changed by editing the values in the `Pspace` and `Initial_value` columns. Simple fixed relationships among parameters may be set by declaring one parameter to have a value like another. In the output above, `GN` is the position of the parameter in the parameter list, `RP_GN` is a related parameter (here the same as `GN` showing parameters are distinct. If, however, a parameter is linked to an earlier parameter, then `RP_GN` for the linked parameter will be the `GN` number of the one it is linked to. The `SCALE` field specifies a fixed scale relationship for the parameter.

VCC statements are used to define the relationships, these 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 units.us(Trait) !LINK 1 2 3 4 5 7 8 12 !SCALE 1 1 2 1 2 1 2 1 2
VCC units.us(Trait) !LINK 6 9 13 # link positions are relative to first 'units' parameter
constrains the matrix to have a pattern with just 5 values represented as
```

A

A 2A

A 2A B

A 2A B C

A 2A B D E

The above constraints from the two `VCC` need to be defined at the end of the model.

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.
#
4, "Variance 1", V, F, 1.0000000, , 4, 1
5, "units.us(Trait);us(Trait)_1", G, P, 0.68444445, , 5, 1.0000
6, "units.us(Trait);us(Trait)_2", G, P, 0.68444445, , 5, 1.0000
7, "units.us(Trait);us(Trait)_3", G, P, 1.3688889, , 5, 2.0000
8, "units.us(Trait);us(Trait)_4", G, P, 0.68444445, , 5, 1.0000
9, "units.us(Trait);us(Trait)_5", G, P, 1.3688889, , 5, 2.0000
10, "units.us(Trait);us(Trait)_6", G, P, 0.68444445, , 10, 1.0000
11, "units.us(Trait);us(Trait)_7", G, P, 0.68444445, , 5, 1.0000
12, "units.us(Trait);us(Trait)_8", G, P, 1.3688889, , 5, 2.0000
13, "units.us(Trait);us(Trait)_9", G, P, 0.68444445, , 10, 1.0000
14, "units.us(Trait);us(Trait)_10", G, P, 0.68444445, , 14, 1
15, "units.us(Trait);us(Trait)_11", G, P, 0.13688889, , 15, 1
16, "units.us(Trait);us(Trait)_12", G, P, 0.68444445, , 5, 1.0000
17, "units.us(Trait);us(Trait)_13", G, P, 0.68444445, , 10, 1.0000
18, "units.us(Trait);us(Trait)_14", G, P, 0.13688889, , 18, 1
19, "units.us(Trait);us(Trait)_15", G, P, 0.68444445, , 19, 1
```

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.

Since `.tsv/.msv/.rsv` files are readily overwritten, if editing one of these files, also change its main name so that **ASReml** does not overwrite it, and specify its name as the argument: `!CONTINUE myparams.msv`, for example.

### 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, by default, 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`
- `FAi` to `CORGH`
- `FAi` to `FAi+1`
- `FAi` to `US`
- `XFAi` to `XFAi+1`
- `XFAi` to `US`
- `US` to `XFA1`, `XFA2`, and `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 reports which `.rsv` files is used. 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)`

## 7.11 Variance model functions available in ASReml

---

- `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)`
- `ar(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 coding could be further reduced 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 7.10 which is located at the end of this chapter for easy access, see Section 7.12. 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 7.10 (`id...matk`) 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...` `xfak`.

In summary

To specify a correlation model, provide the variance model function name given in

- Table 7.10, 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, Section 7.11.6), and singular relationship matrices described in Chapter 9.

If an estimated matrix comes too close to being singular, ASReml will stop iterating.

Let  $\mathbf{x}^T \mathbf{A} \mathbf{x}$  represent an arbitrary quadratic form for  $\mathbf{x} = (x_1, \dots, x_n)^T$ . The quadratic form is said to be nonnegative definite if  $\mathbf{x}^T \mathbf{A} \mathbf{x} \geq 0$  for all  $\mathbf{x} \in \mathbf{R}^n$ . If  $\mathbf{x}^T \mathbf{A} \mathbf{x}$  is nonnegative definite and in addition the null vector  $\mathbf{0}$  is the only value of  $\mathbf{x}$  for which  $\mathbf{x}^T \mathbf{A} \mathbf{x} = 0$ , then the quadratic form is said to be positive definite. Hence the matrix  $\mathbf{A}$  is said to be positive definite if  $\mathbf{x}^T \mathbf{A} \mathbf{x}$  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 7.10

- 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 `chol $k$`  models  $\Sigma = \mathbf{LDL}^T$  where  $\mathbf{L}$  is lower triangular with ones on the diagonal,  $\mathbf{D}$  is diagonal and  $k$  is the number of non-zero off diagonals in  $\mathbf{L}$ .
- In `chol $k$ c` models  $\Sigma = \mathbf{LDL}^T$  where  $\mathbf{L}$  is lower triangular with ones on the diagonal,  $\mathbf{D}$  is diagonal and  $k$  is the number of non-zero sub diagonal columns in  $\mathbf{L}$ . This is somewhat similar to the factor analytic model.
- In `antek` models  $\Sigma^{-1} = \mathbf{UDU}^T$  where  $\mathbf{U}$  is upper triangular with ones on the diagonal,  $\mathbf{D}$  is diagonal and  $k$  is the number of non-zero off diagonals in  $\mathbf{U}$ .
- The `chol $k$`  and `antek` 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, such as

$$\begin{bmatrix} \sigma_{11} & \sigma_{21} & \sigma_{31} \\ \sigma_{21} & \sigma_{22} & \sigma_{32} \\ \sigma_{31} & \sigma_{32} & \sigma_{33} \end{bmatrix}$$

that is, initial values are given in the order,  $1 = \sigma_{11}, 2 = \sigma_{21}, 3 = \sigma_{22}, \dots$

- The `US` model is associated with several special features of `ASReml`. There is a 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 `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 class which accommodates geometric anisotropy and a choice of metrics for random fields observed in two dimensions. This extension, described in detail in Haskard (2006), is given by

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

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

$$\rho_M(d; \phi, \nu) = \{2^{\nu-1} \Gamma(\nu)\}^{-1} \left(\frac{d}{\phi}\right)^\nu K_\nu\left(\frac{d}{\phi}\right), \quad (7.1)$$

where  $\phi > 0$  is a range parameter,  $\nu > 0$  is a smoothness parameter,  $\Gamma(\cdot)$  is the gamma function,  $K_\nu(\cdot)$  is the modified Bessel function of the third kind of order  $\nu$  (Abramowitz and Stegun, 1965,

section 9.6) and  $d$  is the distance defined in terms of  $X$  and  $Y$  axes

$$\begin{aligned} h_x &= x_i - x_j; \\ h_y &= y_i - y_j; \\ s_x &= \cos(\alpha)h_x + \sin(\alpha)h_y; s_y = \sin(\alpha)h_x - \cos(\alpha)h_y; \\ d &= (\delta|s_x|^\lambda + |s_y|^\lambda/\delta)^{1/\lambda}. \end{aligned}$$

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

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

When  $\nu = 1.5$  then

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

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

The final metric parameter  $\lambda$  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  $\nu$  is often set a-priori (Kammann and Wand, 2003, Diggle *et al.*, 2003). However, Stein (1999) and Haskard (2006) demonstrate that  $\nu$  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  $(\phi, \nu = 0.5, \delta = 1, \alpha = 0, \lambda = 2)$ . For example, if we wish to fit a Matérn model with only  $\phi$  estimated and the other parameters set at their defaults then we use **MAT1**. **MAT2** allows  $\nu$  to be estimated or fixed at some other value, for example

```
mat2(fac(xcoord,ycoord) !INIT 0.2 1.0 !GPF)
```

The parameters  $\phi$  and  $\nu$  are highly correlated so it may be better to manually cover a grid of  $\nu$  values.

We note that there is non-uniqueness in the anisotropy parameters of this metric  $d(\cdot)$  since inverting  $\delta$  and adding  $\frac{\pi}{2}$  to  $\alpha$  gives the same distance. This non-uniqueness can be removed by 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  $\alpha$  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. This qualifier is used when you have a factor but do not have the coordinates associated with the factor classes. If the coordinates are in the data, then they can be used directly.

- For one dimensional cases, either
  - `expv(fac(xcoord))` where `X` is the data variable containing the plot positions
  - `expv(Trait !COORD x)` where `x` is a vector of positions for the levels of `Trait`.
- In two dimensions (`iexp()`, `igau()`, `ieuc()`, `aexp()`, `agau()`, `matn()`)

For a **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

ASReml 4.3 has the structures `fak()`, `facv()`, `xfak()`, `rrk()` and `rrdk()`, which are different parameterizations of the factor analytic model in which  $\Sigma$  is modelled as  $\Sigma = \Gamma\Gamma^T + \Psi$  where  $\Gamma^{(\omega \times k)}$  is a matrix of loadings on the covariance scale and  $\Psi$  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  $\Psi$  may not include zeros except in the `xfak()` and `rrk()` formulations
- In the XFA form some or all of the diagonal elements of  $\Psi$  may be zero and in the RR (reduced rank) form, all of the diagonal elements of  $\Psi$  are zero
- Constraints are required in  $\Gamma$  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  $\omega(\omega+1)/2$
- `fak()` and `facvk()` models are rarely used.

In **FAk** models the variance-covariance matrix  $\Sigma^{(\omega \times \omega)}$  is modelled on the correlation scale as  $\Sigma =$

**DCD**, where

- $\mathbf{D}^{(\omega \times \omega)}$  is diagonal such that  $\mathbf{DD} = \text{diag}(\mathbf{\Sigma})$
- $\mathbf{C}^{(\omega \times \omega)}$  is a correlation matrix of the form  $\mathbf{FF}^T + \mathbf{E}$  where  $\mathbf{F}^{(\omega \times k)}$  is a matrix of loadings on the correlation scale and  $\mathbf{E}$  is diagonal and is defined by difference
- The parameters are specified in the order *loadings for each factor (F) followed by the variances* ( $\text{diag}(\mathbf{\Sigma})$ ); when  $k$  is greater than 1, constraints on the elements of  $\mathbf{F}$  are required, see Table 7.5.

FACV $k$  models (CV for *covariance*) are an alternative formulation of FA models in which  $\mathbf{\Sigma}$  is modelled as  $\mathbf{\Sigma} = \mathbf{\Gamma\Gamma}^T + \mathbf{\Psi}$  where  $\mathbf{\Gamma}^{(\omega \times k)}$  is a matrix of loadings on the covariance scale and  $\mathbf{\Psi}$  is diagonal. The parameters in FACV

- Are specified in the order *loadings ( $\mathbf{\Gamma}$ ) followed by variances ( $\mathbf{\Psi}$ )*; when  $k$  is greater than 1, constraints on the elements of  $\mathbf{\Gamma}$  are required, see Table 7.5
- Are related to those in FA by  $\mathbf{\Gamma} = \mathbf{DF}$  and  $\mathbf{\Psi} = \mathbf{DED}$ .

XFA $k$  (X for *extended*) is the third form of the factor analytic model and has the same parameterisation as for FACV, that is,  $\mathbf{\Sigma} = \mathbf{\Gamma\Gamma}^T + \mathbf{\Psi}$ . However, XFA models

- Have parameters specified in the order  $\text{diag}(\mathbf{\Psi})$  and  $\text{vec}(\mathbf{\Gamma})$ ; when  $k$  is greater than 1, constraints on the elements of  $\mathbf{\Gamma}$  are required, see Table 7.5,
- May not be used in  $\mathbf{R}$  structures
- Return the factors as well as the effects
- Permit some elements of  $\mathbf{\Psi}$  to be fixed to zero
- Are computationally faster than the FACV formulation for large problems when  $k$  is much smaller than  $\omega$ .

With multiple factors, some constraints are required to maintain identifiability. 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 formally 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 iterative procedure and after each iteration to update parameters.

The `xfak()` variance structure model function can fit reduced rank variance structures by setting specific variances to zero.

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 initialises the factor  $k + 1$  at  $\sqrt{(\mathbf{\Psi} * 0.2)}$ , 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  $k + 1$  initially aims to explain variation previously incorporated in  $\mathbf{\psi}$ . 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 to the loading part of the **AI** matrix, see `!AIPENALTY` Table 5.8.

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.

`RRk` (RR for *reduced rank*) is the fourth form of the factor analytic model. It is formally just an alternate specification of `xfak()` but with the specific variances all set to zero. A new form of this function, `rrk()`, is available which internally sets all the specific variances ( $\Psi$ ) to zero. `rrk()` is formally just a synonym for `xfak()` which means that if you explicitly supply initial values, you need to supply the  $\Psi$  values as zero.

An almost equivalent way of fitting an `xfak()` structure is to fit an `rrk()` term and a `diag()` term, for example

```
rrk(trial).entry + diag(trial).entry
```

in place of

```
xfak(trial).entry
```

`RRDk` (RR for *reduced rank* and D for *diagonal*) is equivalent to `xfak()`, but fits it as two independent model terms: an `rrk()` term and a `diag()` term. That is writing

```
rrdk(trial).grm(entry)
```

is shorthand (and expanded) to

```
rrk(trial).grm(entry) + diag(trial).grm(entry)
```

This reorganization of the mixed model equations runs faster with large relationship matrices.

## 7.12 Variance models available in ASReml

Table 7.10: Details of the variance functions available in ASReml

| variance<br>structure name             | description | algebraic form | number of parameters <sup>†</sup> |           |           |
|--|-------------|----------------|-----------------------------------|-----------|-----------|
| function name                          |             |                | corr.                             | hom. var. | het. var. |
| <b>correlation models</b>              |             |                |                                   |           |           |
| <b>one-dimensional, equally spaced</b> |             |                |                                   |           |           |

## 7.12 Variance models available in ASReml

| variance<br>structure name | description   | algebraic form   | number of parameters <sup>†</sup> |           |              |
|----------------------------|---|--|-----------------------------------|-----------|--------------|
| function name              |   |  | corr.                             | hom. var. | het. var.    |
| <b>ID</b>                  |   |  |                                   |           |              |
| id                         | identity  | $C_{ii} = 1, C_{ij} = 0, i \neq j$   | 0                                 | 1         | $\omega$     |
| <b>AR1</b>                 |   |  |                                   |           |              |
| ar1                        | 1 <sup>st</sup> order autoregressive                    | $C_{ii} = 1, C_{i+1,i} = \phi_1$<br>$C_{ij} = \phi_1 C_{i-1,j}, i > j + 1$<br>$ \phi_1  < 1$   | 1                                 | 2         | $1 + \omega$ |
| <b>AR2</b>                 |   |  |                                   |           |              |
| ar2                        | 2 <sup>nd</sup> order autoregressive                    | $C_{ii} = 1,$<br>$C_{i+1,i} = \phi_1 / (1 - \phi_2)$<br>$C_{ij} = \phi_1 C_{i-1,j} + \phi_2 C_{i-2,j}, i > j + 1$<br>$ \phi_1  < (1 - \phi_2),  \phi_2  < 1$   | 2                                 | 3         | $2 + \omega$ |
| <b>AR3</b>                 |   |  |                                   |           |              |
| ar3                        | 3 <sup>rd</sup> order autoregressive                    | $C_{ii} = 1, \Omega = 1 - \phi_2 - \phi_3 (\phi_1 + \phi_3),$<br>$C_{i+1,i} = (\phi_1 + \phi_2 \phi_3) / \Omega,$<br>$C_{i+2,i} = (\phi_1 (\phi_1 + \phi_3) + \phi_2 (1 - \phi_2)) / \Omega,$<br>$C_{ij} = \phi_1 C_{i-1,j} + \phi_2 C_{i-2,j} + \phi_3 C_{i-3,j}, i > j + 2$<br>$ \phi_1  < (1 - \phi_2),  \phi_2  < 1,  \phi_3  < 1$ | 3                                 | 4         | $3 + \omega$ |
| <b>SAR</b>                 |   |  |                                   |           |              |
| sar1                       | symmetric<br>autoregressive                             | $C_{ii} = 1,$<br>$C_{i+1,i} = \phi_1 / (1 + \phi_1^2 / 4)$<br>$C_{i+1,i} = \phi_1 / (1 + \phi_1^2 / 4)$<br>$C_{ij} = \phi_1 C_{i-1,j} - \phi_1^2 / 4 C_{i-2,j}, i > j + 1$<br>$ \phi_1  < 1$   | 1                                 | 2         | $1 + \omega$ |
| <b>SAR2</b>                |   |  |                                   |           |              |
| sar2                       | constrained<br>autoregressive 3 used<br>for competition | as for AR3 using<br>$\phi_1 = \gamma_1 + 2\gamma_2,$<br>$\phi_2 = -\gamma_2(2\gamma_1 + \gamma_2),$<br>$\phi_3 = \gamma_1\gamma_2^2,$  | 2                                 | 3         | $2 + \omega$ |
| <b>MA1</b>                 |   |  |                                   |           |              |

## 7.12 Variance models available in ASReml

| variance<br>structure name | description                          | algebraic form   | number of parameters <sup>†</sup> |                                    |   |
|----------------------------|--------------------------------------|--|-----------------------------------|------------------------------------|---|
| function name              |                                      |  | corr.                             | hom. var.                          | het. var.                               |
| <b>MA1</b>                 |                                      |  |                                   |                                    |   |
| ma1                        | 1 <sup>st</sup> order moving average | $C_{ii} = 1,$<br>$C_{i+1,i} = \theta_1 / (1 + \theta_1^2)$<br>$C_{ji} = 0, j > i + 2$<br>$ \theta_1  < 1$<br>Some authors replace the moving<br>average $\theta_i$ by $-\theta_i$  | 1                                 | 2                                  | $1 + \omega$                            |
| <b>MA2</b>                 |                                      |  |                                   |                                    |   |
| ma2                        | 2nd order<br>moving average          | $C_{ii} = 1,$<br>$C_{i+1,i} = -\theta_1 / (1 - \theta_2) / (1 + \theta_1^2 + \theta_2^2)$<br>$C_{i+2,i} = -\theta_2 / (1 + \theta_1^2 + \theta_2^2)$<br>$C_{ji} = 0, j > i + 2$<br>$\theta_2 \pm \theta_1 < 1$<br>$ \theta_1  < 1,  \theta_2  < 1$ | 2                                 | 3                                  | $2 + \omega$                            |
| <b>ARMA</b>                |                                      |  |                                   |                                    |   |
| arma                       | autoregressive<br>moving average     | $C_{ii} = 1,$<br>$C_{i+1,i} = (\theta - \phi)(1 - \theta\phi)(1 + \theta^2 - 2\theta\phi)$<br>$C_{ji} = \phi C_{j-1,i}, j > i + 1$<br>$ \theta  < 1,  \phi  < 1$   | 2                                 | 3                                  | $2 + \omega$                            |
| <b>CORU</b>                |                                      |  |                                   |                                    |   |
| coru                       | uniform correlation                  | $C_{ii} = 1, C_{ij} = \phi, i \neq j$  | 1                                 | 2                                  | $1 + \omega$                            |
| <b>CORB</b>                |                                      |  |                                   |                                    |   |
| corb                       | banded correlation                   | $C_{ii} = 1$<br>$C_{i+j,i} = \phi_j, 1 \leq j \leq \omega - 1$<br>$ \phi_{ij}  < 1$  | $\omega - 1$                      | $\omega$                           | $2\omega - 1$                           |
| <b>CORG</b>                |                                      |  |                                   |                                    |   |
| corg                       | general<br>CORGH=US correlation      | $C_{ii} = 1$<br>$C_{ij} = \phi_{ij}, i \neq j$<br>$ \phi_{ij}  < 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

| variance<br>structure name         | description           | algebraic form   | number of parameters <sup>†</sup> |           |              |
|------------------------------------|-----------------------|--|-----------------------------------|-----------|--------------|
| function name                      |                       |  | corr.                             | hom. var. | het. var.    |
| one-dimensional unequally spaced   |                       |  |                                   |           |              |
| IEXP                               |                       |  |                                   |           |              |
| iexp                               | exponential           | $C_{ii} = 1,$<br>$C_{ij} = \phi^{ x_i - x_j }, i \neq j$<br>$x_i$ are <i>coordinates</i><br>$0 < \phi < 1$                       | 1                                 | 2         | $1 + \omega$ |
| GAU                                |                       |  |                                   |           |              |
| gau                                | gaussian              | $C_{ii} = 1,$<br>$C_{ij} = \phi^{(x_i - x_j)^2}, i \neq j$<br>$x_i$ are <i>coordinates</i><br>$0 < \phi < 1$                     | 1                                 | 2         | $1 + \omega$ |
| two-dimensional irregularly spaced |                       |  |                                   |           |              |
|                                    |                       | $\mathbf{x}$ and $\mathbf{y}$ vectors of coordinates<br>$\theta_{ij} = \min(d_{ij}/\phi_1, 1)$<br>$d_{ij}$ is euclidean distance |                                   |           |              |
| IEXP                               |                       |  |                                   |           |              |
| iexp                               | isotropic exponential | $C_{ii} = 1,$<br>$C_{ij} = \phi^{ x_i - x_j  +  y_i - y_j }, i \neq j$<br>$0 < \phi < 1$   | 1                                 | 2         | $1 + \omega$ |
| IGAU                               |                       |  |                                   |           |              |
| igau                               | isotropic gaussian    | $C_{ii} = 1,$<br>$C_{ij} = \phi^{(x_i - x_j)^2 + (y_i - y_j)^2}, i \neq j$<br>$0 < \phi < 1$                                     | 1                                 | 2         | $1 + \omega$ |
| IEUC                               |                       |  |                                   |           |              |
| ieuc                               | isotropic euclidian   | $C_{ii} = 1,$<br>$C_{ij} = \phi^{\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}}, i \neq j$<br>$0 < \phi < 1$                              | 1                                 | 2         | $1 + \omega$ |
| LVR                                |                       |  |                                   |           |              |
| lvr                                | linear variance       | $C_{ij} = (1 - \theta_{ij})$<br>$0 < \phi_1$   | 1                                 | 2         | $1 + \omega$ |

## 7.12 Variance models available in ASReml

| variance structure name              | description  | algebraic form   | number of parameters <sup>†</sup> |           |                                |
|--------------------------------------|--|--|-----------------------------------|-----------|--------------------------------|
| function name                        |  |  | corr.                             | hom. var. | het. var.                      |
| <b>SPH</b>                           |  |  |                                   |           |                                |
| sph                                  | spherical  | $C_{ij} = 1 - \frac{3}{2}\theta_{ij} + \frac{1}{2}\theta_{ij}^3$ $0 < \phi_1$  | 1                                 | 2         | $1 + \omega$                   |
| <b>CIR</b>                           |  |  |                                   |           |                                |
| cir                                  | circular (Webster & Oliver, 2001, p113)                              | $C_{ij} = 1,$ $-\frac{2}{\pi}(\theta_{ij}\sqrt{1 - \theta_{ij}^2} + \sin^{-1} \theta_{ij})$ $0 < \phi < 1$   | 1                                 | 2         | $1 + \omega$                   |
| <b>AEXP</b>                          |  |  |                                   |           |                                |
| aexp                                 | anisotropic exponential  | $C_{ii} = 1,$ $C_{ij} = \phi_1^{ x_i - x_j } \phi_2^{ y_i - y_j }$ $0 < \phi_1 < 1, 0 < \phi_2 < 1$  | 2                                 | 3         | $2 + \omega$                   |
| <b>AGAU</b>                          |  |  |                                   |           |                                |
| 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         | $2 + \omega$                   |
| <b>MAT<math>k</math></b>             |  |  |                                   |           |                                |
| mat $k$                              | Matérn with first $1 \leq k \leq 5$ parameters specified by the user | $C_{ij} = \text{Matérn: see text}$ $\phi > 0 \text{ range, } \nu \text{ shape}(0.5)$ $\delta > 0 \text{ anistropy ratio}(1)$ $\alpha \text{ anistrophy angle}(0)$ $\lambda(1 2) \text{ metric}(2)$ | $k$                               | $k+1$     | $k + \omega$                   |
| <b>heterogeneous variance models</b> |  |  |                                   |           |                                |
| <b>DIAG</b>                          |  |  |                                   |           |                                |
| diag                                 | diagonal = IDH idh   | $\Sigma_{ii} = \phi_i \Sigma_{ij} = 0, i \neq j$   | -                                 | -         | $\omega$                       |
| <b>US</b>                            |  |  |                                   |           |                                |
| us                                   | unstructured general covariance matrix                               | $\Sigma_{ij} = \phi_{ij}$  | -                                 | -         | $\frac{\omega(\omega + 1)}{2}$ |
| <b>OWN<math>k</math></b>             |  |  |                                   |           |                                |

## 7.12 Variance models available in ASReml

| variance structure name | description                |  | algebraic form  | number of parameters <sup>†</sup> |           |                              |
|-------------------------|----------------------------|--|---|-----------------------------------|-----------|------------------------------|
| function name           |                            |  |   | corr.                             | hom. var. | het. var.                    |
| <code>ownk</code>       | user                       | explicitly forms $\mathbf{V}$ and $\partial\mathbf{V}$ |   | -                                 | -         | $k$                          |
| <b>ANTE1</b>            | 1 <sup>st</sup>            | $k$ order  | $\Sigma^{-1} = \mathbf{U}\mathbf{D}\mathbf{U}^\top$   | -                                 | -         | $\frac{\omega(\omega+1)}{2}$ |
| <code>ante1</code>      | $k^{\text{th}}$            | antependence   | $D_{ii} = d_i, D_{ij} = 0, i \neq j$  |                                   |           |                              |
| <b>ANTEk</b>            | $1 \leq k \leq \omega - 1$ |  | $U_{ii} = 1, U_{ij} = u_{ij}, 1 \leq j - i \leq k$  |                                   |           |                              |
| <code>ante k</code>     |                            |  | $\mathbf{U}_{ij} = 0, i > j$  |                                   |           |                              |
| <b>CHOL1</b>            | 1 <sup>st</sup>            | $k$ order  | $\Sigma = \mathbf{L}\mathbf{D}\mathbf{L}^\top$  | -                                 | -         | $\frac{\omega(\omega+1)}{2}$ |
| <code>chol1</code>      | $k^{\text{th}}$            | cholesky   | $D_{ii} = d_i, D_{ij} = 0, i \neq j$  |                                   |           |                              |
| <b>CHOLk</b>            | $1 \leq k \leq \omega - 1$ |  | $L_{ii} = 1, L_{ij} = l_{ij}, 1 \leq i - j \leq k$  |                                   |           |                              |
| <code>chol k</code>     |                            |  |   |                                   |           |                              |
| <b>FA1</b>              | 1 <sup>st</sup>            | $k$ order  | $\Sigma = \mathbf{D}\mathbf{C}\mathbf{D}$   | -                                 | -         | $\omega + \omega$            |
| <code>fa1</code>        | $k^{\text{th}}$            | factor analytic  | $\mathbf{C} = \mathbf{F}\mathbf{F}^\top + \mathbf{E}$   |                                   |           | $k\omega + \omega$           |
| <b>FAk</b>              |                            |  | $\mathbf{F}$ contains $k$ correlation factors   |                                   |           |                              |
| <code>fa k</code>       |                            |  | $\mathbf{E}$ diagonal   |                                   |           |                              |
|                         |                            |  | $\mathbf{D}\mathbf{D} = \text{diagonal}(\Sigma)$  |                                   |           |                              |
| <b>FACV[1]</b>          | 1 <sup>st</sup>            | $k$ order  | $\Sigma = \mathbf{\Gamma}\mathbf{\Gamma}^\top + \mathbf{\Psi}$                                | -                                 | -         | $\omega + \omega$            |
| <code>facv1</code>      | $k^{\text{th}}$            | factor analytic covariance form                        | $\mathbf{\Gamma}$ contains covariance factors   |                                   |           | $k\omega + \omega$           |
| <b>FACVk</b>            |                            |  | $\mathbf{\Psi}$ contains specific variance  |                                   |           |                              |
| <code>facv k</code>     |                            |  |   |                                   |           |                              |
| <b>XFA1</b>             | 1 <sup>st</sup>            | $k$ order  | $\Sigma = \mathbf{\Gamma}\mathbf{\Gamma}^\top + \mathbf{\Psi}$                                | -                                 | -         | $\omega + \omega$            |
| <code>xfa1</code>       | $k^{\text{th}}$            | extended factor analytic                               | $\mathbf{\Gamma}$ contains covariance factors   |                                   |           | $k\omega + \omega$           |
| <b>XFAk</b>             |                            |  | $\mathbf{\Psi}$ contains specific variance  |                                   |           |                              |
| <code>xfa k</code>      |                            |  |   |                                   |           |                              |
| <b>RR1</b>              | 1 <sup>st</sup>            | $k$ order  | $\Sigma = \mathbf{\Gamma}\mathbf{\Gamma}^\top$  | -                                 | -         | $\omega + \omega$            |
| <code>rr1</code>        | $k^{\text{th}}$            | random regression                                      | $\mathbf{\Gamma}$ contains covariance factors   |                                   |           | $k\omega + \omega$           |
| <b>RRk</b>              | $k^{\text{th}}$            | reduced rank factor analytic                           | $\mathbf{\Psi}$ contains specific variances set to zero from <b>XFAk</b> (see Section 7.11.6) |                                   |           |                              |
| <code>rr k</code>       |                            |  |   |                                   |           |                              |

## 7.12 Variance models available in ASReml

| variance<br>structure name                  | description  | algebraic form  | number of parameters <sup>†</sup> |           |           |
|---|--|---|-----------------------------------|-----------|-----------|
| function name                               |  |   | corr.                             | hom. var. | het. var. |
| <b>RRD<math>k</math></b><br><br><i>rrdk</i> |  | $\Sigma = \Gamma\Gamma^T + \Psi$<br>As with <b>XFA<math>k</math></b> but split into two<br>terms: <i>rrk</i> + diag |                                   |           |           |
| <b>relationship matrices<sup>‡</sup></b>    |  |   |                                   |           |           |
| <b>NRM</b><br><i>nrn</i>                    | relationship matrix derived from pedigree                            |   | 0                                 | 1         | -         |
| <b>GINV1</b><br><i>giv1</i>                 | generalized relationship inverse matrix number 1                     |   | 0                                 | 1         | -         |
| $\vdots$                                    | $\vdots$   |   | $\vdots$                          | $\vdots$  | $\vdots$  |
| <b>GINV<math>k</math></b><br><i>givk</i>    | generalized relationship inverse matrix $k$ ,<br>$k = 1, \dots, 498$ |   | 0                                 | 1         | -         |
| <b>GRM1</b><br><i>grm1</i>                  | generalized relationship number 1                                    |   | 0                                 | 1         | -         |
| $\vdots$                                    | $\vdots$   | $\vdots$  | $\vdots$                          | $\vdots$  |           |
| <b>GRM<math>k</math></b><br><i>grmk</i>     | generalized relationship matrix $k$ ,<br>$k = 1, \dots, 498$         |   | 0                                 | 1         | -         |

<sup>†</sup> This is the number of variance structure parameters,  $\omega$  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*.

<sup>‡</sup> 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 fiber 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
subject * !=V0
wolfrat.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 wethers 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 fiber diameter (microns). The code (`wether.as`) to the right performs a basic bivariate analysis of this data.

```
Orange Wether Trial 1984-1988
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
:
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 – see Section 5.4.1).
- *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` *i.e.*  $nt$  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

$$I_n \otimes \Sigma$$

where  $\Sigma^{(t \times t)}$  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 `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)
```

- If provided, the initial values are for the lower triangle of the (symmetric) matrix specified row-wise.

```
Orange Wether Trial 1984-1988
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
```

- 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, *i.e.* it is to be positive definite matrix.

The special qualifiers relating to multivariate analysis are `!ASUV` and `!ASMV t`, see Table 5.7 for details

- To use an error structure other than **US** for the residual stratum you must also specify `!ASUV` (see Table 5.7) 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 5.7).
- `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**<sup>1</sup> 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*, *parent\_1*, *parent\_2*) appears as the first three fields of the data file, the data file can double as the pedigree file.

```
Pedigree file example
Animal !P
Sire !A
Dam
Line 2
AgeOfDam
adailygain
Y2
Y3
harvey.ped !ALPHA
harvey.dat
adailygain ~ mu Line,
!r nrmv(Animal !INIT 0.25)
residual idv(units)
```

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.

---

<sup>1</sup> [https://jvanderw.une.edu.au/Mixed\\_Models\\_for\\_Genetic\\_analysis.pdf](https://jvanderw.une.edu.au/Mixed_Models_for_Genetic_analysis.pdf)



## 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 3 192 390 224
102 Sire_1 0 1 3 154 403 265
103 Sire_1 0 1 4 185 432 241
104 Sire_1 0 1 4 183 457 225
105 Sire_1 0 1 5 186 483 258
106 Sire_1 0 1 5 177 469 267
107 Sire_1 0 1 5 177 428 271
108 Sire_1 0 1 5 163 439 247
109 Sire_2 0 1 4 188 439 229
110 Sire_2 0 1 4 178 407 226
111 Sire_2 0 1 5 198 498 197
112 Sire_2 0 1 5 193 459 214
113 Sire_2 0 1 5 186 459 244
114 Sire_2 0 1 5 175 375 252
115 Sire_2 0 1 5 171 382 172
116 Sire_2 0 1 5 168 417 275
117 Sire_3 0 1 3 154 389 238
118 Sire_3 0 1 4 184 414 246
119 Sire_3 0 1 5 174 483 229
120 Sire_3 0 1 5 170 430 230
:
```

## 9.4 Reading in the pedigree file

The syntax for specifying a pedigree file in the **ASReml** command file is

*pedigree\_file* [*qualifiers*] [*pedigree modification qualifiers*]

- The *qualifiers* 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, the variance model function name `nrm()`.
- The inverse relationship matrix is written to `ainverse.bin`.
  - 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 in Section [16.13](#).

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<br>!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 $A^{-1}$ ), 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 <i>basename.aif</i> .   |
| !CSKIP <i>c</i>    | this qualifier instructs ASReml to ignore the first <i>c</i> columns of the pedigree file. A pedigree file typically has 3 or 4 fields being the identifiers for the individual: its Sire, its Dam and maybe its sex or inbreeding value ( <i>f</i> ). This qualifier is intended to facilitate reading a data file as a pedigree file when the file has <i>c</i> other fields at the beginning of each line.   |
| !FGEN [ <i>f</i> ] | 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 <i>f</i> 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 field of the pedigree, will be assigned an inbreeding coefficient <i>f</i> .   |
| !GIV<br>!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 <i>basename_Parent.ped</i> and the diagonal elements of the A-inverse to <i>basename_Q.giv</i> with offspring identifiers. If !GROUPS is also specified, this .giv file will include the !GROUPSDF qualifier on its first line.   |
| !GOFFSET <i>o</i>  | An alternative to group constraints (see !GROUPS below) is to shrink the group effects by adding the constant <i>o</i> ( $> 0$ ) to the diagonal elements of $A^{-1}$ pertaining to groups. When a constant is added, no adjustment of the degrees of freedom is made for genetic groups.<br><br>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 <i>g</i>   | includes genetic groups in the pedigree. The first <i>g</i> 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.<br><br>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 Table 5.8) is designed to help as without it, reordering may associate singularities in the $A$ matrix with random effects which at the very least is confusing. When the $A$ 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. |

## 9.4 Reading in the pedigree file

| qualifier             | description   |
|-----------------------|---|
| !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 $A^{-1}$ 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 $A^{-1}$ in ASReml was based on Quaas (1976).  |
| !REPEAT               | tells ASReml to ignore repeat occurrences of lines in the pedigree file.<br><b>Warning</b> Use of this option will avoid the check that animals occur in generational order, but generational order is still required.  |
| !SARGOLZAEI           | an alternative procedure for computing $A^{-1}$ was developed by Sargolzaei <i>et al.</i> (2005).   |
| !SELF <i>s</i>        | 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 <i>s</i> and from outcrossing with probability (1- <i>s</i> ). 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.  |
| !SAVEGIV [ <i>f</i> ] | This qualifier is used to write the $A^{-1}$ matrix in binary form so that it can be read back in ASReml and so save re-computation and have reduced reading time. If <i>f</i> = 3, the inverse matrix is written as a binary .sgiv file in single precision, and if <i>f</i> = 4 writes the inverse matrix as a binary .dgiv file in double precision. The default value is <i>f</i> = 3.  |
| !SKIP <i>n</i>        | allows you to skip <i>n</i> 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. Genetic groups with no members will be dropped by the !SORT process. In an effort to save pre-processing effort, if <i>pedigreefile</i> is specified as <i>basename</i> .SRT and this file already exists, ASReml will assume the sorting has already been performed to create the file and ignore !SORT. However if <i>basename</i> .SRT does not exist, this sorted file will be created from the file <i>basename</i> whether or not !SORT is included.              |
| !UPPER                | all lower-case characters are converted to upper case. This qualifier was introduced for the case of a pedigree where names had not been recorded consistently with respect to case.  |
| !XLINK                | 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 $A^{-1}$ 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 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. |

It is possible to read a relationship matrix (or its inverse) directly from a file rather than producing it from the pedigree file, further details are presented in Section 9.7. 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, two 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 the inverse 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
...
...      grml(Mline) nrm(Fline)
```

The qualifier !SAVEGIV can be added to the pedigree file line to write  $A^{-1}$  as a sparse binary .sgiv file. To read back  $A^{-1}$  as a  $G^{-1}$  in a subsequent run, several changes will be required to the command line coding (farther details in Section 9.7.4). For example, if the original job (say PED.as) included lines

```
Animal !P
...
Pedigree.csv !SAVEGIV !DIAG
Y ~ ... !r nrm(Animal)
```

Copy it as say GIV.as and change it to say:

```
Animal !A !L PED.aif # Animal !P
...
Pedigree_A.sgiv      # Pedigree.csv !SAVEGIV !DIAG
Y ~ ... !r grml(Animal) # Y ~ ... !r nrm(Animal)
```

## 9.5 Pedigree modification qualifiers

Pedigree pre-processing has been extended to allow removal of unnecessary individuals: those with no data or descendants with data, sometimes called trimming, and recursive removal of base individuals (individuals with unknown parents) that have no data and only one offspring, sometimes called pruning. Both these operations do not change the likelihood or parameter estimates but can save computation. There is the facility to also limit the number of generations of ancestors in the pedigree. This again can save computation but can change estimates as it implicitly assumes the ancestors left out have no genetic variation. There is an option to form and use a sparse inverse relationship matrix just on a specified set of individuals.

## 9.6 Genetic groups

The extended syntax is of the form

*Pedigreefile* [qualifiers] [`!TRIM` *filename* [*field*] [`!SKIP` *k*] [`!NOPRUNE`] [`!KEEP` *g*] [`!REDUCE`]

with the optional qualifiers `!SKIP`, `!NOPRUNE`, `!KEEP`, and `!REDUCE` appearing after the `!TRIM` qualifier. The pedigree modification qualifiers are listed and described in Table 9.2.

Table 9.2: Pedigree modification qualifiers

| qualifier  | description   |
|--|---|
| <code>!TRIM</code><br><i>filename</i> [ <i>field</i> ] | constructs a trimmed pedigree. Individuals with data are identified from <i>field</i> (default 1) of <i>filename</i> (typically the data file). It does not check to whether any particular response variable has a response for that record. The qualifier <code>!KEEP g</code> identifies the number of generations of ancestors to include. The pedigree is then pruned. The modified pedigree is written to the file <i>pedigreefile</i> .TRM. The pedigree lines are flagged either <code>!RETAIN</code> or <code>!REMOVE</code> to distinguish those in <i>filename</i> from their ancestors. Genetic groups will generally be lost under trimming. |
| <code>!SKIP k</code>                                   | skips the first <i>k</i> lines of <i>filename</i> . Note that this qualifier should occur after <i>filename</i> and that to skip lines of <i>pedigreefile</i> the <code>!SKIP</code> qualifier should appear before <code>!TRIM</code> .  |
| <code>!NOPRUNE</code>                                  | instructs ASReml not to prune the trimmed pedigree.   |
| <code>!KEEP g</code>                                   | identifies the number of generations of ancestors to include (without <code>!KEEP</code> ) all ancestors are included.  |
| <code>!REDUCE</code>                                   | specifies that the inverse relationship matrix is based on just individuals with data. The flags <code>!RETAIN</code> and <code>!REMOVE</code> in the trimmed pedigree file are used to identify individuals with data. Generally this will make the reduced inverse relationship matrix much denser and so is only useful in special situations where family size is small or the number of ancestors retained is small (definitely less than 10,000).   |

## 9.6 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,
!r nrm(Animal !INIT 0.25))
residual idv(units)
```

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

```
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
```

## 9.7 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.[b|d|r|s]grm [qualifiers]`

`name.[b|d|r|s]giv [qualifiers]`

The other qualifiers for reading **GIV/GRM** files are detailed in Table 9.3 and they are discussed in the following sections.

The content and layout of the file is implied by the filename extension. The extension should end in `giv` if the matrix is inverted, `grm` if not inverted. **ASReml** will usually need to use the inverse **GRM** and will invert the supplied matrix if required. The file is assumed ASCII if the file extension is `.giv` or `.grm`. It is assumed binary if the file extension is `.bgiv`, `.bgrm`, `.dgiv`, `.dgrm`, `.rgiv`, `.rgrm`, `.sgiv`, or `.sgrm`. Details of binary file structures are given in Section 9.7.8.

Table 9.3: Qualifiers for reading GIV/GRM files grouped by functionality

| qualifier/file extension  | description                             | section |
|---|---|---------|
| <code>!SKIP <i>n</i></code>   | skipping lines in ASCII files           | 9.7     |
| <code>!LDET <i>r</i></code>   | setting log determinant                 | 9.7     |
| <code>!GROUPSDF <i>n</i></code>   | incorporating genetic groups            | 9.7.1   |
| <code>[!ND !PSD !NSD][!PRECISION [<i>n</i>]][!ADD <i>d</i> [!NONULL]]</code>    | dealing with singularities              | 9.7.3   |
| <code>!SAVEGIV [<i>f</i>]</code>  | saving <b>GIV</b>                       | 9.7.4   |
| <code>!HINV <i>f</i> [!HSKIP <i>h</i>] [!OMEGA <i>ω</i>] [!TAU <i>τ</i>]</code> | forming the <b>H</b> matrix             | 9.7.5   |
| <code>!KEEPGRM</code>   | keeping the <b>GRM</b> matrix in memory | 9.7.6   |
| <code>!EIGTRANSFORM</code>  | transforming model                      | 9.7.7   |
| <code>.bgiv, .bgrm, .sgiv, .sgrm, .dgiv, .dgrm, .rgiv, .rgrm</code>             | binary file layouts                     | 9.7.8   |

The `!SKIP n` qualifier applies to ASCII files to control how many leading lines to ignore.

ASCII files are read free format (space or comma separated) in either dense (row-wise) or sparse (cell-wise) layout. ASReml determines which from the first line.

- A dense format file has the matrix presented row by row with each row beginning on a new line, and may contain the lower triangle part of the full row; **ASReml** just reads the row-wise triangular part. If the file contains row/column labels, as for example in the file `MyGRM` exporting a GRM matrix from **R** with `write.table(MYGRM< `My.grm')``, these labels will be recognized and then ignored.
- 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.

```

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 first line of a `.giv` file may have `!LDET det` appended to save **ASReml** recalculating the log determinant value.

The `!LDET det` qualifier may be given after the file name of a GIV file if not given in the file.

The `.giv` file presented in the code box gives the **G** inverse matrix below

$$\begin{bmatrix} I_4 & \mathbf{0} \\ \mathbf{0} & I_4 \otimes \begin{bmatrix} 1.067 & -0.267 \\ -0.267 & 1.267 \end{bmatrix} \end{bmatrix}$$

**G** (inverse) files must be specified on the line(s) immediately prior to the data file line after any pedigree file.

- Up to 498 **G** (inverse) matrices may be defined. Using the variance model function `grmk(f)`; the variance structure of the factor  $f$  can be associated to the  $k$ th **GRM** (or equivalently associated with the inverse of the  $k$ th **GIV**, for example

```
grmlv(animal !INIT 0.12)
```

or

```
coruh(site).giv2(variety)
```

The number and order of the rows must agree with the size and order of the associated factor.

**Warning** **ASReml** does not hold the list of genotype names as an attribute of the **G** matrix and so cannot automatically align the data order with the  $\mathbf{G}^{-1}$  matrix order as **ASReml-R** does. Therefore, genotype names must be supplied, in the  $\mathbf{G}^{-1}$  matrix order, when defining the data factor which will be associated with the **G** matrix.

```
genotype !A !L Gorder.txt
```

instructs **ASReml** to code the variable genotype in the order of level names in the first field of ASCII file `Gorder.txt`. Otherwise, genotype is coded in the order level names are encountered in the data file, which may not match the **G** matrix.

The **GRM** file (and hence `Gorder.txt`) may include genotypes not in the data file; fitting `grm(genotype)` in the model will predict effects for all the genotypes. If the data file includes genotypes not in the **GRM** file (*i.e.* not in `Gorder.txt`), these are appended to the genotype list and the  $\mathbf{G}^{-1}$  matrix is extended with an identity matrix to cover those extra genotypes. When the job is run, check `genotype` has the correct number of levels and correct order in the `.sln` file.

### 9.7.1 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 the number of constraints.

## 9.7 Reading a user defined (inverse) relationship matrix

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 A inverse for this pedigree was saved, it will contain !GROUPSDF 3 in the GIV file.

### 9.7.2 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,  $0.5I_{74}$ . This model is simply an example which is easy to verify.

Note that harvey.giv is specified on the line immediately preceding harvey.dat.

```
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,
!r grmlv(Sire !INIT 0.25)
residual idv(units)
```

```
1 1 .5
2 2 .5
3 3 .5
4 4 .5
5 5 .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.

### 9.7.3 Dealing with singularities

The qualifiers [!ND|!PSD|!NSD] [!PRECISION[n]] [!ADD d [!NONULL]] provide options for inverting a GRM file and discussed in Table 9.4 below. They are ignored if the inverted file is supplied.

Table 9.4: GRM inversion qualifiers

| qualifier               | action   |
|-------------------------|--|
| !ND   !PSD   !NSD       | <b>ASReml</b> uses a most efficient matrix inverter if none of these qualifiers are used. It uses a less efficient inverter if !ND is specified, and again if !PSD/NSD is specified.<br><br>Linear dependencies occur, for example, when the list of individuals includes clones. Rows with zero elements occur when the <b>GRM</b> 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 <b>ASReml</b> 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. |
| !PRECISION [ <i>n</i> ] | changes the value used to declare a singularity when inverting a <b>GRM</b> file from 1E-10 to 1E- <i>n</i> ( <i>n</i> > 3). A default value of 7 is used for <i>n</i> if it is not set. The !PRECISION qualifier also allows the use of Lagrangian multipliers to accommodate linear dependencies to allow matrices with singularities to run.  |
| !ADD <i>d</i>           | adds a constant 0.000001 * <i>d</i> to non-zero diagonal elements of the <b>GRM</b> matrix supplied.   |
| !NONULL                 | in conjunction with !ADD <i>d</i> to add 0.000001 * <i>d</i> to all diagonal elements. This is intended for when the matrix elements are not of sufficient precision resulting in the matrix not being positive definite.  |

### 9.7.4 Saving the G inverse

If !SAVEGIV [*f*] is specified, the inverse of the **GRM** matrix is written to a file written as a lower triangular row-wise format. If *f* = 1, 2 or -1 the matrix inverse is written to a .giv file. With *f* = 1 the sparse (cellwise *row col value*) format is used and with *f* = 2 the dense (row-wise *value(1:row)*) format is used. If *f* = -1 **ASReml** decides the format depending if the number of non-zero elements is greater than (dense) or less than (sparse) half the number of elements in the matrix. If *f* = 3 the inverse matrix is written as a binary .sgiv file in single precision and *f* = 4 writes the inverse matrix as a binary .dgiv file in double precision. The default value of *f* is 3. The written file also includes on the first line !LDET *r* where *r* is the log determinant of the grm matrix used in the computation of the log-likelihood. The qualifier !LDET *r* with a .giv file specification allows the user to specify the log determinant of the corresponding .grm file if the user has constructed the .giv file independently of **ASReml**. This can save computation time.

### 9.7.5 Forming the H matrix

The **H** (or hybrid) matrix is a particular form of a **G** matrix obtained by merging an **A** (numerator relationship) matrix with a **G** (genomic relationship) matrix pertaining to a subset of the genotypes in **A** (Legarra *et al.*, 2009). The matrix is of the form

$$H_{\tau, \omega}^{-1} = A^{-1} + \begin{pmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & (\tau G^{-1} - \omega A_{22}^{-1}) \end{pmatrix}$$

where  $A_{22}^{-1}$  is the inverse of the portion of the **A** matrix that contains the genotyped individuals. Note that the cells present in the **H** inverse can be tuned by  $\omega$  and  $\tau$ , which have default values of 1 (for more details see Martini *et al.* 2018).

The qualifiers

```
!HINV GRM_ID_file.txt [!HSKIP h] [!OMEGA  $\omega$ ] [!TAU  $\tau$ ]
```

on a GRM definition line forms the special **G** inverse known as an **H** inverse. A pedigree from which to form an **A** inverse must have already been specified. The genotype identifiers in the **G** matrix are to be specified as a list in the ASCII file provided as the argument to the **!HINV** qualifier. All identifiers must also be present in the pedigree.

For example

```
!ARG 1 !RENAME
H inverse Example
ID !P
GID !A !L GID.txt
yld
pedigree.csv !SKIP 1
Gmatrix.grm !HINV GID.txt
data.csv !SKIP 1
yld ~ mu !r grm2(ID)
```

where `pedigree.csv` contains the pedigree for the **A** matrix, `Gmatrix.grm` (or `.giv`) holds the **G** matrix (or **G**<sup>-1</sup>), and `!HINV GID.txt` instructs **ASReml** to form the **H** inverse. `GID.txt` is a file containing the list of genotype identifiers for the **G** matrix, which must be a subset of the pedigree file identifiers. Use `!HSKIP h` to skip header lines in `GID.txt`, `!OMEGA  $\omega$`  and `!TAU  $\tau$`  are optional tuning parameters discussed above. If the **H** matrix has been formed outside of **ASReml**, it can be read as a **GRM** matrix.

In the example, in order to form the **H**<sup>-1</sup> matrix, the pedigree file from which the **A**<sup>-1</sup> matrix is first formed, then the **G** (or its inverse) matrix is specified and read, and finally, the **!HINV** qualifier forms the **H**<sup>-1</sup> matrix. Note that the **GRM** line with the **!HINV** qualifier, defines two inverse matrices available for use in the model: the **G**<sup>-1</sup> and the **H**<sup>-1</sup>. Therefore, we could fit any of `nrm(ID)`, `grm1(GID)` or `grm2(ID)` in the model. This assumes that the levels associated with the **!HINV** qualifier are included in the pedigree.

**ASReml** saves the **H**<sup>-1</sup> matrix as a binary file (filename given in the output) which can subsequently be used directly (saving the setup time in the subsequent runs).

### 9.7.6 Keeping the GRM in memory

Below are some additional arguments to use when handling GRM matrices that provide some additional memory and processing efficiencies.

`!KEEPGRM` instructs **ASReml** to keep in memory the **GRM** matrix as well as its inverse.

`!KEEPGRM` is needed when fitting a model term like

```
sat (Env) .grmk (Gen)
```

(replacing `diag (Env) .grm (Gen)` usually fitted in combination with `rr1 (Env) .grmk (Gen)`).

The `sat ()` form allows the associated model term (`grmk (Gen)`) to apply just to the genotypes with data in the particular `Env`, greatly reducing the size of the model. To set up the **G** inverse for each level of `Env`, **ASReml** needs to access the uninverted **GRM**. When say *Environments* are nested within *Fields*, the individual inverse **GRM** matrices can be formed on the basis of genotypes in a field by writing to fit a separate model term for each environment and writing

```
sat (Env !VGROUP Field) .grm (Gen)
```

The !VPGROUP qualifier can be used to constrain variance parameters in the associated components to common values (see Section 7.3.3).

## 9.7.7 Eigen-Model transformation !EIGTRANSFORM

In the context of needing to fit a large genomic *Animal* model to many traits, an eigenvector transformation of the model design matrix can result in faster processing (for more details see Lee and van der Werf 2016). This idea is relevant when the GRM term dominates the model. It essentially turns the part of coefficient matrix relating to genomic effects to a diagonal structure at the expense of turning the (much fewer) non-genomic effects into dense equations.

The !EIGTRANSFORM qualifier on the GRM line invokes a singular value decomposition (SVD) of  $G$  ( $G = UDU'$  with  $UU' = I$  and  $D$  diagonal), holds the  $U'$  and  $D$  for subsequent use and writes them to files (...\_D.sgrm, ..., \_U.sgrm). If the  $U'$  and  $D$  files already exist,  $U'$  and  $D$  are retrieved from a file. Then, if this GRM is specified in the model, and the model and data meet the necessary requirements, the model design matrix is transformed (pre-multiplied by  $U'$ ) except for the columns specific to the GRM;  $D$  is used as the GRM variance matrix for the genetic effects in the analysis of the transformed model so that this block of the  $C$  matrix remains diagonal. Generally, this will run much faster.

The eigen-transformation is only possible when there is one data record for each genotype, as occurs with the *Animal* model. A large amount of workspace is required for the process of transforming the design matrix.

The fitted effects between the two models agree except for the genomic BLUPs. The BLUPs from the conventional BLUP ( $u$ ) are calculated as  $u = Us$ , where  $s$  are the BLUPs from the transformed analysis. ASReml 4.3 does not calculate them at present.

For example, the common genomic animal model

```
!WORK 6
10K bivariate data
ID !A !LL20 !L
CG * CGs *           # contemporary group factors
imf sf5               # response variates
A.sgiv !GDENSE        # A inverse genomic matrix
data.csv !SKIP 1      # data file in same order as A22
imf ~ CG !r grml(ID)
```

The equivalent model and two other similar equivalent models can be fitted with

```
!WORK 6 !RENAME 1 !ARG 1 2 3 !DOPART $1
10K bivariate data
ID !A !LL20 !L
CG * CGs *           # contemporary group factors
imf sf5               # response variates
tCG !G 376           # CG design matrix
A.grm !EIGTRANSFORM   # instruction for eigenvalue analysis
data.csv !SKIP 1      # data file
```

## 9.7 Reading a user defined (inverse) relationship matrix

```
!PART 1
imf ~ CG !r grml(ID)      # transformed model
!PART 2
sf5 ~ CG !r grml(ID)      # transformed model
!PART 3
imf sf5 ~ Trait !r us(Trait).grml(ID) !f Trait.CG # bivariate transf. model
```

The `!EIGTRANSFORM` qualifier performs an SVD (Singular Value Decomposition, eigen analysis) of the **G** matrix and holds the eigenvalues and vectors. It also flags that when the model is fitted, the design matrix is to be transformed using the eigenvectors.

This approach only works when the data file has the same (or fewer) rows as the **GRM** matrix and the number of other effects in the model is substantially less than the number of genotypes. It will save considerable time when there are many response variates with no missing values to analyse, especially for bivariate analyses.

Comparing the two models given in the code above, with 9,688 genotypes, the conventional univariate analysis took 9 sec (to invert **GRM**) + 8×43 sec (for 8 iterations). The overhead of the SVD factorization was 160 sec and 8 sec (for transforming the model); the transformed analysis took 8×2 sec for 8 iterations. A conventional bivariate analysis took 8×313 sec. The transformed bivariate analysis took 8×13 sec, a considerable difference.

The fitted effects between the two models agree except for the genomic BLUPs. As indicated before, the BLUPs from the conventional BLUP (**u**) are calculated as  $\mathbf{u} = \mathbf{Us}$  where **s** are the BLUPs from the transformed analysis.

### 9.7.8 Binary G matrices

ASReml was first developed to fit *animal* models using the pedigree-based numerator relationship matrix (**NRM**). The inverse **NRM** ( $\mathbf{A}^{-1}$ ) is sparse. Later, the dense genomic relationship matrix (**GRM**) was devised and ASReml was extended to fit models using such matrices as supplied by the user, or derived directly from a marker matrix (search **GRR** discussed elsewhere).

ASReml 4.3 can read in a **GRM** matrix in various forms defined by content (**G** or  $\mathbf{G}^{-1}$ ), form (ASCII, REAL\_S or REAL\_R) and layout (row-wise or cell-wise). These options are now discussed. Their content and form are indicated by the following filename extensions:

Table 9.5: GRM/GIV file extensions

| content           | ASCII | REAL_S         | REAL_D | REAL_R |
|-------------------|-------|----------------|--------|--------|
| <b>G</b>          | .grm  | .sgrm<br>.bgrm | .dgrm  | .rgrm  |
| $\mathbf{G}^{-1}$ | .giv  | .sgiv<br>.bgiv | .dgiv  | .rgiv  |

The binary forms are preferred because the files are smaller and accuracy is greater. Providing the inverse and log-determinant of the matrix saves processing time calculating them. REAL\_S refers to the **FORTRAN** sequential binary file structure in which each 'record' is enclosed in a 'wrapper' indicating the record size in bytes. When ASReml writes a binary file, by default it uses the single precision REAL\_S file structure. REAL\_D is a double precision binary file but the extra precision

## 9.7 Reading a user defined (inverse) relationship matrix

is generally unnecessary. REAL\_R refers to the R(C) binary form which does not include any 'record size' information.

Typically,  $\mathbf{G}$  and  $\mathbf{G}^{-1}$  are dense matrices ((nearly) all cells non-zero) and are half-stored. However, some very large matrices have significant sparsity (most cells are zero).

When ASReml writes files with these extensions, it knows how to read the file even though the internal layout may vary depending on whether the file is essentially dense or sparse.

Some common layouts are now discussed with respect to  $\mathbf{R}$  and an NRM matrix of order 10 based on the pedigree shown to the right.

GRM matrices in ASReml are half-stored row-wise in either dense or sparse layouts. The dense layout, for  $NR$  (number of rows), means each of  $NR*(NR+1)/2$  cells are explicitly stored row-wise and so a particular cell  $I, J$  can be found at position  $I*(I-1)/2 + J$  where  $I \geq J$ . The sparse layout collapses the vector of values, dropping (non-diagonal) values of zero. It uses a second vector, parallel to the values vector to specify the column ( $J$ ) values for each cell; row ( $I$ ) values are implicit in the order of values with the diagonal cells always retained.

| ID | Sire | Dam |
|----|------|-----|
| 1  | 0    | 0   |
| 2  | 0    | 0   |
| 3  | 0    | 0   |
| 4  | 1    | 1   |
| 5  | 1    | 1   |
| 6  | 2    | 2   |
| 7  | 4    | 6   |
| 8  | 5    | 6   |
| 9  | 7    | 8   |
| 10 | 9    | 9   |

### ASCII half-stored $\mathbf{G}^{-1}$ matrix cell-wise

Processing the small pedigree shown above, with the !SAVEGIV qualifier to form the inverse relationship matrix, creates the ped\_A.giv file shown right. It contains qualifiers for the log determinant and the number of genetic groups. The first field is the row number, the second is the column number and the third is the matrix cell value.

ASReml can read this .giv file back in as a  $\mathbf{G}^{-1}$  matrix.

```
!LDET -6.6130181 !GROUPSDF 0
1 1 5.000000000
2 2 3.000000000
3 3 1.000000000
4 1-2.000000000
4 4 3.000000000
5 1-2.000000000
5 5 3.000000000
:
9 7-1.000000000
9 8-1.000000000
9 9 4.909090909
10 9-2.909090909
10 10 2.909090909
```

### ASCII half-stored $\mathbf{G}^{-1}$ matrix row-wise

The information in the .giv file was used to create a matrix ( $\mathbf{A}^{-1}$ ) in R which was inverted to form an NRM matrix. The following code in R

```
> NRM <- solve(Ainv)
> write(round(NRM,5), 'NRM.grm')
```

Produces the file NRM.grm containing

|      | "v1" | "v2" | "v3" | "v4"  | "v5"  | "v6" | "v7"   | "v8"   | "v9"   | "v10"   |
|------|------|------|------|-------|-------|------|--------|--------|--------|---------|
| "1"  | 1    | 0    | 0    | 1     | 1     | 0    | 0.5    | 0.5    | 0.5    | 0.5     |
| "2"  | 0    | 1    | 0    | 0     | 0     | 1    | 0.5    | 0.5    | 0.5    | 0.5     |
| "3"  | 0    | 0    | 1    | 0     | 0     | 0    | 0      | 0      | 0      | 0       |
| "4"  | 1    | 0    | 0    | 1.5   | 1     | 0    | 0.75   | 0.5    | 0.625  | 0.625   |
| "5"  | 1    | 0    | 0    | 1     | 1.5   | 0    | 0.5    | 0.75   | 0.625  | 0.625   |
| "6"  | 0    | 1    | 0    | 0     | 0     | 1.5  | 0.75   | 0.75   | 0.75   | 0.75    |
| "7"  | 0.5  | 0.5  | 0    | 0.75  | 0.5   | 0.75 | 1      | 0.625  | 0.8125 | 0.8125  |
| "8"  | 0.5  | 0.5  | 0    | 0.5   | 0.75  | 0.75 | 0.625  | 1      | 0.8125 | 0.8125  |
| "9"  | 0.5  | 0.5  | 0    | 0.625 | 0.625 | 0.75 | 0.8125 | 0.8125 | 1.3125 | 1.3125  |
| "10" | 0.5  | 0.5  | 0    | 0.625 | 0.625 | 0.75 | 0.8125 | 0.8125 | 1.3125 | 1.65625 |



ASReml can read this and three variations back in as a **G** matrix: without labels, without elements above diagonal and without both. Both the layouts cell-wise and row-wise can be used for **G** and **G**<sup>-1</sup> files, according to the file extension.

If the !LDET qualifier is omitted on a .giv file, ASReml will calculate the log-determinant. And, the !GROUPSDF qualifier is only needed when the **G**<sup>-1</sup> matrix is actually an **A**<sup>-1</sup> formed with genetic groups.

### REAL\_S BINARY half-stored files (.bgiv, .bgrm, .sgiv, .sgrm)

Usually, these files will be formed by ASReml, and ASReml can read them back, the details are not important.

### REAL\_R BINARY half-stored files (.rgiv, .rgrm)

ASReml can also read binary files formed using the R writeBin() function (or by a C program). Unlike the FORTRAN sequential binary files, these have no record markers and all values are 32bit real values. .rgiv files need the header to specify the log determinant.

Given a matrix called GRM held in R, it can be written to a binary file that ASReml can read by the R code shown to the right.

If writing the inverse GRM matrix in R, use the filename extension .rgiv, and include a header line in the file by inserting the R code line

```
writeBin(c(NR, 0, Ldet), Tfile, size=4)
```

```
NR <- dim(GRM)[1] # dimension
Tfile <- file("My.rgrm", "wb")
for (i in 1:NR)
{
  writeBin(GRM[1:i,i], Tfile,
size=4)
}
close(Tfile)
```

after the Tfile line, where Ldet is the log determinant of the GRM matrix usually obtained while inverting the GRM matrix.

For a sparse stored matrix held in a data frame SAI as described above, use the R code shown to the right.

Note that

- .sgiv and .sgrm files are binary, single precision 'sequential' files containing sparse or dense matrices of various styles. ASReml examines the file to determine the style. The allowed styles include those written by ASReml and binary versions of the ASCII layouts.
- .rgiv and .rgrm files are 'C-style' binary, single precision, written by R as binary versions for the ASCII layouts. R code to create an .rgrm file is

```
NR <- dim(GRM)[1] # Get dimension
Tfile <- file("My.rgrm", "wb")
for (i in 1:NR) {
  writeBin (GRM[1:i,i], Tfile, size = 4)
```

```
SAI <- read.table('ped_A.giv', skip = 1)
NV <- dim(SAI)[1] # Get length
NR <- SAI[NV,1]
Tfile <- file("SAI.rgiv", "wb")
writeBin(c(NR, 0, -6.61302), Tfile, size = 4)
for (i in 1:NV) {
  writeBin(c(SAI[i,2], SAI[i,3]), Tfile, size = 4)
}
close(Tfile)
```



```
}  
close(Tfile)
```

- If writing the **inverse** of a GRM in R, use the filename extension `.rgiv`, and include a header line in the file by inserting the R code line

```
writeBin (c(NR, 0, Ldet), Tfile, size = 4)
```

After the `Tfile` line, where `Ldet` is the log determinant of the GRM matrix usually obtained while inverting the GRM matrix.

- `.dgiv` and `.dgrm` files are binary, dense format double precision files.
- `.bgiv` and `.bgrm` files are treated as `.sgiv` and `.sgrm` files.

## 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  $\mathbf{u}$ , a vector of  $f$  factor effects with  $\mathbf{v}$  a vector of  $m$  regression effects through the model  $\mathbf{u} = \mathbf{M}\mathbf{v}$  where the matrix  $\mathbf{M}$  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  $\mathbf{Z}\mathbf{M}\mathbf{v}$  where  $\mathbf{Z}$  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  $\mathbf{Z}\mathbf{u}$  with a variance structure for  $\mathbf{u}$  based on  $\mathbf{M}\mathbf{M}'$ . **ASReml** can read the matrix  $\mathbf{M}$  associated with a factor and group of regressor variables from a `.grr` file, construct a GRM matrix ( $\mathbf{G} = \mathbf{M}\mathbf{M}'/s$  with  $s$  a scaling term), fit the equivalent model and report both factor and regressor predictions. One common case of this model is when  $\mathbf{u}$  represents genotype effects, the regressors represent SNP marker counts (typically 0/1/2 representing allele counts and 0 and 2 representing homozygotes) and  $\mathbf{v}$  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 [ $f$ ] Regressors [ $m$ ] [qualifiers]
```

where

*M.grr* is the name of the file to be 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, but setting *f* ensures adequate space is allocated; **ASReml** will count the actual number.

*Regressors* is the name for the set of regressor variables.

## 9.8 Factor effects with large Random Regression Models

$m$  sets the number of regressor variables (default is the number of names found); must be set if there are extraneous fields to be ignored.

The qualifiers are described in [Table 9.6](#), [Table 9.7](#), and [Table 9.8](#).

**Table 9.6: Main GRR line qualifiers**

| qualifier  | action   |
|------------|--|
| !CSKIP $c$ | indicates $c$ fields are to be skipped before the factor identifiers are read.   |
| !NOID      | indicates that the factor identifiers are not present in the .grr file.  |
| !CSKIP $c$ | indicates $c$ fields are to be skipped before the regressor variables are read.  |
| !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. |
| !DOMINANCE | creates a marker-based dominance matrix ( $\mathbf{G_D}$ ) as defined below.   |
| !EPISTATIC | creates the epistatic matrices as Hadamard products of $\mathbf{G_A}$ and $\mathbf{G_D}$ as defined below.   |
| !SKIP $s$  | specifies how many lines are to be skipped before reading the regressor data.  |

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 presented in [Table 9.7](#).

**Table 9.7: Additional GRR line qualifiers**

| qualifier        | action  |
|------------------|---|
| !KEEPGRM         | instructs ASReml to keep in memory the GRM matrix as well as its inverse.   |
| !SAVEGIV [ $f$ ] | instructs ASReml to write the $\mathbf{G}^{-1}$ matrix in binary form so that it can be read back, and so save re-computation and have reduced reading time. This facility has been extended to pedigree and GRR lines. If $f=3$ the inverse matrix is written as a binary .sgiv file in single precision, and if $f=4$ writes the inverse matrix as a binary .dgiv file in double precision. The default value is $f=3$ . If used with !KEEPGRM on the GIV/GRM lines, the uninverted $\mathbf{G}$ matrix is also written to .sgrm/ .dgrm binary files. |
| !PSD $s$         | declares that the derived variance matrix may have up to $s$ singularities.   |

## 9.8 Factor effects with large Random Regression Models

| qualifier   | action  |
|-------------|---|
| !PEV        | requests calculation of Prediction Error Variance of marker effects which are reported in the .mef file. Calculation of Prediction error variances was computationally very expensive. The algorithm has been drastically improved and the recommendation is to always use !PEV.              |
| !CENTRE [c] | requests ASReml to centre the regressors at $c$ if $c$ is specified else at the individual regressor means; otherwise the $\mathbf{G}$ matrix is formed from uncentered regressors. Note that centering introduces a singularity in the $\mathbf{G}$ matrix and !PSD $s$ 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.

Table 9.8: Rarely used GRR line qualifiers

| qualifier    | action  |
|--------------|---|
| !SMODE $b$   | sets the storage mode for the regressor data, since marker files maybe very large but marker data may be simple (0/1/2).<br><br>$b = 2$ 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 centered marker data to scale the $\mathbf{MM}'$ matrix so that it is a genomic matrix.  |

An example is presented below

```
Nassau Clone Data
Nfam 71 !A
Nfemale 26 !A
Nmale 37 !A
Clone 860 !A
MatOrder
rep 8 !A
iblk 80 !A
tree
row
col
prop 1 !A
culture 2 !A
treat 2 !A
measure 1 !A
SURV
DBH6
HT6
HT8
CWAC6 !M-9
snpData.grr Clone !SKIP 1 !HEAD 0 !CENTRE !MARKERS 4854 !IDS 923 !PEV
nassau_cut_v3.csv !MAXIT 30 !SKIP 1 !GDENSE
HT6 ~ mu culture culture.rep !r grml(Clone) idv(Clone) rep.iblk
```

## 9.8 Factor effects with large Random Regression Models

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...
...
547853,2,2,1,2,2,2,1,2,2,0,2,1,2,2,2,2,2,2,1,2,...
547966,2,2,1,1,1,2,0,2,2,1,2,2,2,2,2,2,2,2,1,2,...
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 860 !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 !SKIP 1 !HEAD 0 !CENTRE          !IDS 923 !PEV
Class names for factor "Clone" are initialized from the .grr file.
Note: SNP data line begins: 140099,2,2,1,2,2,2,2,2,2,1,2,1,2,1,1,2
Note: Markers coded -9 treated as missing.
      Use !RANGE min max  if this value is to be included.
      1  25  58|,-9,2,2,2
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
Regressors centered at 1.00
      Sigma (2p(1-p)) is      1057.12558

GRM  Identifier      Rows      Type      LogDet  GroupsDF
   1  snpData.grr      923        9      -947.89        0
QUALIFIERS: !MAXIT 30 !SKIP 1 !GDENSE
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
   5  MatOrder           0      0      55.00    486.6     914.0     247.9
   6  rep              8      0      0         1     4.4837       8
   7  iblk           80      0      0         1    40.1164      80
   8  tree            0      0      1.000     7.473     14.00     4.018
   9  row             0      0      1.000    28.52     56.00    16.09
```

## 9.8 Factor effects with large Random Regression Models

```

10 col                0      0  1.000      10.50      20.00      5.760
Warning: Fewer levels found in prop than specified
11 prop                2      0      1      1.0000      1
12 culture             2      0      1      1.4945      2
13 treat               2      0      1      1.4945      2
Warning: Fewer levels found in measure than specified
14 measure             2      0      1      1.0000      1
15 SURV                0      6  1.000      0.9991      1.000      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
Note: The GRM matrix specified in grml(Clone) is smaller ( 923) than Clone ( 926)
      and is extended with an Identity to cover the extra levels.
22 grml(Clone)          926
23 idv(Clone)           926
24 rep.iblk             640 6 rep : 8 7 iblk : 80

```

Note: GRM structure seems larger than model factor for term grml(Clone)

Note: Random model term grml(Clone) is included in the DENSE equations.

Use !GDENSE -1 before model line to cancel this action.

Forming 2511 equations: 945 dense.

Initial updates will be shrunk by factor 0.316

\* This job uses all of the 4 processor threads. \*

Note: 11 singularities detected in design matrix.

|         |          |     |        |         |        |        |        |
|---------|----------|-----|--------|---------|--------|--------|--------|
| 1 LogL= | -32843.2 | S2= | 8954.4 | 6391 df | 0.1000 | 0.1000 | 0.1000 |
| 2 LogL= | -32796.3 | S2= | 8566.5 | 6391 df | 0.1288 | 0.1372 | 0.1169 |
| 3 LogL= | -32755.9 | S2= | 8130.3 | 6391 df | 0.1811 | 0.1993 | 0.1373 |
| 4 LogL= | -32740.5 | S2= | 7843.1 | 6391 df | 0.2404 | 0.2626 | 0.1471 |
| 5 LogL= | -32738.1 | S2= | 7709.7 | 6391 df | 0.2798 | 0.3037 | 0.1485 |
| 6 LogL= | -32738.1 | S2= | 7699.2 | 6391 df | 0.2814 | 0.3078 | 0.1490 |
| 7 LogL= | -32738.1 | S2= | 7699.1 | 6391 df | 0.2813 | 0.3078 | 0.1491 |

Final parameter values 0.2813 0.3078 0.1491

- - - Results from analysis of HT6 - - -

Akaike Information Criterion 65484.18 (assuming 4 parameters).

Bayesian Information Criterion 65511.23

Coefficient of Determination: NDF 7.00 DenDF 606.13 Fall 399.91 CD 82.20

Approximate stratum variance decomposition

| Stratum           | Degrees-Freedom | Variance | Component | Coefficients |
|-------------------|-----------------|----------|-----------|--------------|
| grml(Clone)       | 563.53          | 28366.0  | 5.8       | 0.1 6.9 1.0  |
| rep.iblk          | 618.31          | 30079.5  | 0.0       | 9.4 0.0 1.0  |
| idv(Clone)        | 280.23          | 15561.4  | 0.0       | 0.0 6.8 1.0  |
| Residual Variance | 4928.93         | 7699.12  | 0.0       | 0.0 0.0 1.0  |

| Model_Term  |            | Gamma    | Sigma   | Sigma/SE | % C |
|-------------|------------|----------|---------|----------|-----|
| grml(Clone) | GRM_V 926  | 0.281307 | 2165.82 | 5.88     | 0 P |
| rep.iblk    | IDV_V 640  | 0.307777 | 2369.61 | 13.00    | 0 P |
| idv(Clone)  | IDV_V 926  | 0.149101 | 1147.95 | 5.94     | 0 P |
| Residual    | SCA_V 6399 | 1.00000  | 7699.12 | 49.64    | 0 P |

Wald F statistics

| Source of Variation | NumDF | DenDF | F_inc   | P_inc |
|---------------------|-------|-------|---------|-------|
| 20 mu               | 1     | 289.7 | 686.41  | <.001 |
| 12 culture          | 1     | 605.8 | 2616.73 | <.001 |
| 21 culture.rep      | 6     | 606.2 | 30.44   | <.001 |

## 9.8 Factor effects with large Random Regression Models

```
Note: The DenDF values are calculated ignoring fixed/boundary/singular
      variance parameters using numerical derivatives.
22 grml(Clone)                926 effects fitted
24 rep.iblk                   640 effects fitted
23 idv(Clone)                 926 effects fitted (    66 are zero)
* This job used at least .3 of the 2.0 Gbyte of primary workspace. *
  78 possible outliers in Section 1: see .res file
Finished:    02 Dec 2025 11:52:52.307    LogL Converged
```

### 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.

| Marker_name | Effect         | SE_Effect | Weighting_in_G |
|-------------|----------------|-----------|----------------|
| 1           | -0.8976960E-01 | 1.430801  | 0.9459614E-03  |
| 2           | -0.2582894     | 1.377880  | 0.9459614E-03  |
| 3           | 0.6248010      | 1.356795  | 0.9459614E-03  |
| 4           | -0.8198767E-01 | 1.353692  | 0.9459614E-03  |
| 5           | 0.9336195      | 1.346462  | 0.9459614E-03  |
| ...         |                |           |                |
| 4850        | 0.5665308      | 1.371614  | 0.9459614E-03  |
| 4851        | 0.1813440      | 1.373347  | 0.9459614E-03  |
| 4852        | 0.2433674E-01  | 1.371125  | 0.9459614E-03  |
| 4853        | -0.3148131     | 1.367573  | 0.9459614E-03  |
| 4854        | 0.6874512E-01  | 1.419857  | 0.9459614E-03  |

- Particular columns of the `.grr` data can be included in the model using the `grr(Factor, i)` model term where `i` specifies which (number) regressor variable to include.

### Dominance and Epistatic GRM matrices for marker (`.grr`) data

GRM matrices are typically formed from marker matrices ( $\mathbf{M}$ ) in which SNPs are coded 0, 1 or 2 and arranged with genotypes in rows (lines of the file) and SNPs as data fields. Vitezica *et al.* (2017) define

$$\mathbf{G}_A = \mathbf{M}\mathbf{M}' / (2 \sum_{j=1}^m p_j(1 - p_j))$$

where  $\mathbf{M}$  has elements  $(0 - 2p_j)$ ,  $(1 - 2p_j)$  and  $(2 - 2p_j)$  for genotypes AA, AB and BB, respectively,  $2p_j$  being the mean for SNP  $j$

$$\mathbf{G}_D = \mathbf{K}\mathbf{K}' / (4 \sum_{j=1}^m p_j^2(1 - p_j^3))$$

where  $\mathbf{K}$  has elements  $-2p_j^2$ ,  $2p_j(1 - 2p_j)$  and  $-2(1 - p_j^2)$  for genotypes AA, AB and BB, respectively. Also

$$\begin{aligned} \mathbf{G}_{AA} &= \mathbf{G}_A \# \mathbf{G}_A / (\text{tr}(\mathbf{G}_A \# \mathbf{G}_A) / n) \\ \mathbf{G}_{AD} &= \mathbf{G}_A \# \mathbf{G}_D / (\text{tr}(\mathbf{G}_A \# \mathbf{G}_A) / n) \\ \mathbf{G}_{DD} &= \mathbf{G}_D \# \mathbf{G}_D / (\text{tr}(\mathbf{G}_D \# \mathbf{G}_D) / n) \end{aligned}$$

where  $\#$  represents the Hadamard product and  $\text{tr}()$  is the trace function.

The matrices are available in **ASReml** with use of the **!DOMINANCE** and **!EPISTATIC** qualifiers on the `.grm` file specification line which supplies the file containing the marker (0, 1, 2) matrix, where **!DOMINANCE** requests  $\mathbf{G}_D$  be formed (along with  $\mathbf{G}_A$ ), and **!EPISTATIC** requests  $\mathbf{G}_{AA}$  be formed (also  $\mathbf{G}_{AD}$  and  $\mathbf{G}_{DD}$  if  $\mathbf{G}_D$  is available).

The matrices formed are listed like

| GRM | Identifier  | Rows | Type | LogDet   | GroupsDF |
|-----|-------------|------|------|----------|----------|
| 1   | snpData.grm | 923  | 9    | -947.89  | 0        |
| 2   | snpData_DOM | 923  | 9    | -225.00  | 0        |
| 3   | snpData_AxA | 923  | 9    | -1377.84 | 0        |
| 4   | snpData_AxD | 923  | 9    | -701.63  | 0        |
| 5   | snpData_DxD | 923  | 9    | -410.54  | 0        |

and selected for including in the model with the `grmk()` function where `k` selects the matrix to use.

## 9.9 Genetic Trimming in MET Models with !KEEPGRM, sat() and !VGROUP

The situation where not all genotypes are observed in all environments is becoming more common. That is, when each environment (`Env`) samples different subsets of the genotypes (`Gen`). This, when combined with the use of a **GRM** matrix to specify the genetic relationships, can quickly result in a large and fairly dense set of equations to solve.

The general variance unstructured (**US**) structure might then be specified as `us(Env).grm(Gen)`. However, since both `us(Env)` and `grm(Gen)` are dense, this variance structure is dense, and therefore slow to fit. Furthermore, with fitting more than six environments, the `us(Env)` structure is likely to be not positive definite and therefore it is difficult to fit directly.

It is common then to use the factor analytic variance structure, `xfak(Env).grm(Gen)`, in place of the `us()` structure. The above computational effort can be greatly reduced by fitting the factor analytic variance structure using the ‘*reduced rank + diag*’ form, for example:

```
rrk(Env).grm(Gen) + diag(Env).grm(Gen)
```

However, further computational improvements are possible. The approach presented here is motivated by the work of Mazur (2021) and concerns the fitting of genotype by environment factor analytic models with a dense `grm(Gen)` matrix term. As mentioned previously, `xfak(Env).grm(Gen)` can be more efficiently fitted as `rrk(Env).grm(Geno) + diag(Env).grm(Geno)`. Mazur (2021) suggested replacing `diag(Env).grm(Gen)` by a set of separate model terms, one for each level of `Env` using only the subset of genotypes with data for that level (*i*) of `Env` and using the associated reduced **GRM** matrices. The author calls this operation *trimming* and this leads to the same variance model (hence, identical variance parameter estimates in the two models).

The expanded model limits the prediction of genotypes to the environments where they occur and this can result in dramatic reduction of computing time, as a result of creating a separate model term for each environment using a reduced environment specific **GRM** matrix. In particular, you may want an average genotype effect (across environments) or relative genotype effects for only those genotypes present at a particular environment.

This operation of forming separate variance structures for each level of environment is analogous to the operation at the residual level of forming similar variance models for each level of `sat()` but the sizes are defined by the data associated with each level of `sat()`. **ASReml 4.3** uses

```
sat (Env) .grm (Gen)
```

to specify this operation. Each of the expanded terms is denoted in the output file as

```
sat (Env, i) .grm (Gen | Env_i)
```

where `i` indexes the environment, `grm (Gen | Env_i)` denotes the reduced **GRM** matrix specific for the genotypes present at environment `i`. With this model, the `rrk (Env) .grm (Gen)` term fits the common (factor) effects and the `sat (Env) .grm (Gen)` terms are the environment specific ‘specific’ variances (genetic deviation from the common effect).

In order to form the reduced **GRM** matrices, the full **GRM** matrix needs to be available and the **GRM** qualifier `!KEEPGRM` is used for this.

In some analyses, experiments are grouped, and the same subset of genotypes are used in all experiments in a group. This means that a smaller set of reduced **GRM** (and **GIV**) need forming. The qualifier `!VGROUP group` has been introduced to allow that, for example

```
sat (Exp !VGROUP Group) .grm (Gen)
```

The nested association between `Exp` and `Group` is deduced from the data and the generated model terms are denoted

```
sat (Exp, i) .grm (Gen | Group_j)
```

where `Gen | Group_j` is the ‘subset’ of genotypes associated with field `j`, and is fitted using a reduced **GRM** pertaining to those genotypes in the data pertaining to experiment `i` which is a member of group `j`.

As an extension to this model, if it is desired that all the specific variance are the same (corresponding to fitting `idv (Env) .grm (Gen)`), the qualifier `!VPGROUP mu` may be specified as in the following example

```
assay ~ mu Env
!r idv (Env !INIT 1) .id (date) grm (Gen INIT 0.37) idv (sel !INIT 0.15),
  id (loc) .grm (Gen !INIT 0.0) .idv (loc.sel !INIT 0.07),
  sat (Env !VPGROUP mu) .grm (Gen)
...
PREDICT Gen
```

An alternative way of making the 191 specific variances equal is to use



```
:
at(env).grm(gen))
VCC sat(Env).grm(Gen) !LINK 1:191
```

where `sat(Env).grm(Gen)` expands to 191 model terms like `at(Env,001).grm(gGn|env_01)` and `grm(Gen|Env_01)` fits the **GRM** to just those (44) entries present in experiment (1). The directive

```
VCC sat(Env).grm(Gen) !LINK 1:191
```

causes them all to have a common variance component.

We can also use `!VPGROUP` to constrain the variance parameters for a parameter reduced version of this model. Consider the untrimmed model

```
!PART 3
Yield ~ mu Exp !r Rep.Exp Block.Rep.Exp grm(Gen),
      Exp.grm(Gen)
```

Three trimmed versions of the model are:

```
!PART 33
Yield ~ mu Exp !r Rep.Exp Block.Rep.Exp grm(Gen),
      sat(Exp !VPGROUP mu).grm(Gen)

!PART 34
Yield ~ mu + Exp !r Rep.Exp Block.Rep.Exp grm(Line),
      sat(Exp).grm(Line)
VCC sat(Env).grm(Line) !LINK 1:24

!PART 35
Yield ~ mu + Exp !r Rep.Exp Block.Rep.Exp grm(Line),
      sat(Exp !VPGROUP GExp).grm(Line)
```

Parts 33 and 34 are exactly equivalent to Part 3, having a common variance

```
sat(Exp).giv(Line)
```

Part 35 uses an (arbitrary) Experiment grouping factor

```
(GExp != Exp !MOD 5 !+1)
```

to demonstrate estimating a common specific variance component for experiments in the same group. There is a more detailed wheatgrass example in Section [11.5.1](#). In this example the trimmed model took 33.3 seconds per iteration compared with 116.7 seconds per iteration for the original model.

# 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` transformation 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 [qualifiers] ~ factors
```

where

- `TABULATE` is the directive name, appearing on a new line.
- *response\_variables* is a list of variates for which means are required.
- *~ 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.

Table 10.1: TABULATE directive qualifiers

| qualifier                              | action   |
|--|--|
| !WT weight                             | nominates a variable containing weights.   |
| !COUNT r                               | requests counts as well as means to be reported.   |
| !DECIMALS [d]<br>( $1 \leq d \leq 7$ ) | requests means be reported with $d$ decimal places. If omitted, <b>ASReml</b> 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.  |
| !PRINTALL                              | reports empty cells in the tabulation. By default, such cells are not reported. This is useful for identifying the empty cells and where they appear in the class list. A '.' is used to represent the various statistics other than count which is zero so the empty classes are easy to see. |

**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 relevant 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 associations 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.

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 a 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) are placed after model/residual lines. The maximum number of cells to be predicted across all predict tables has been increased to 1,000,000.

The syntax is

PREDICT *factors* [*qualifiers*]

where

```
NIN Alliance Trial 1989
  variety !A
:
  column 11
nin89.asd !SKIP 1
yield ~ mu variety !r
idv(repl)
PREDICT variety
```

- 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 10.2, 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 Table 5.6) 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 – see Section 5.4.1) 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 `!DEFINE`, `!GROUP`, `!SUBGROUP`, `!SUBSET` and `!MBF` are implicitly defined through their base variables and cannot 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 sets of variables are included in the classify set (defined by using `!G`), 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.

Table 10.2 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.

Table 10.2: List of prediction qualifiers

| qualifier   | action  |
|---|---|
| <b>controlling formation of tables</b>  |   |
| <code>!ASSOCIATE [v]</code>   | 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.   |
| <code>!AVERAGE f [weights]</code><br><code>!AVERAGE f 'file' [, n]</code>     | <p>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 <code>!AVERAGE</code>, <code>!ASSOCIATE</code> or <code>!PRESENT</code> qualifiers.</p> <p>Explicit weights may be supplied directly or from a file. The default is equal weights.</p> <p><i>weights</i> can be expressed like <code>{3*1 0 2*1}/5</code> to represent the sequence 0.2 0.2 0.2 0.2 0.2. The string inside the curly brace is expanded first and the expression <math>n*c</math> means <math>n</math> occurrences of <math>c</math>.</p> <p>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 '<math>n</math>' is specified in which case they are taken from field/column <math>n</math>.</p> |
| <code>!ASAVERAGE f [weights]</code><br><code>!ASAVERAGE f 'file' [, n]</code> | <p>is used to control averaging over associated factors. The default is to simply average at the base level. Hierarchical averaging is achieved by listing the associated factors to average in <math>f</math>.</p> <p>Explicit weights may be supplied directly or from a file as for <code>!AVERAGE</code>.</p>   |
| <code>!PARALLEL [v]</code>  | 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.   |
| <code>!PRESENT v</code>   | <p>is used when averaging is to be based only on cells with data. <math>v</math> is a list of variables and may include variables in the classify set. <math>v</math> may not include variables with an explicit <code>!AVERAGE</code> qualifier. The variable names in <math>v</math> 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. <b>ASReml</b> works out what combinations are present from the design matrix. It may have trouble with complicated models such as those involving <code>and()</code> terms.</p> <p>A second <code>!PRESENT</code> qualifier is allowed on a <code>PREDICT</code> statement (but not with <code>!PRWTS</code>). The two lists must not overlap.</p>   |
| <code>!PRWTS v</code>   | is used in conjunction with the first <code>!PRESENT v</code> list to specify the weights that <b>ASReml</b> will use for averaging that <code>!PRESENT</code> table. More details are given below.   |

## 10.3 Prediction

| qualifier                                   | action   |
|---|--|
| <b>controlling inclusion of model terms</b> |  |
| <code>!EXCEPT <i>t</i></code>               | causes the prediction to include all fitted model terms not in <i>t</i> .  |
| <code>!IGNORE <i>t</i></code>               | causes <b>ASReml</b> to set up a prediction model based on the default rules and then removes the terms in <i>t</i> . This might be used to omit the spline Lack of fit term ( <code>!IGNORE fac(x)</code> ) from predictions as in<br><pre>yield ~ mu x variety !r spl(x) fac(x) PREDICT x !IGNORE fac(x)</pre> which would predict points on the spline curve averaging over <code>variety</code> .  |
| <code>!ONLYUSE <i>t</i></code>              | causes the prediction to include only model terms in <i>t</i> . It can be used for example to form a table of slopes as in<br><pre>HI ~ mu X variety X.variety PREDICT variety X 1 !ONLYUSE X X.variety</pre>  |
| <code>!PAC <i>t</i></code>                  | (Predict Actual Coefficients) is used instead of <code>!ONLYUSE <i>t</i></code> when <i>t</i> is a term which is a design function of a variable (like <code>leg()</code> , <code>mbf()</code> or <code>spl()</code> ), and the prediction pertains to the actual coefficients, not the function of the coefficients, as in<br><pre>!MBF mbf(times,12) MyZeds.txt !SKIP 1 !KEY 1 !RENAME Zeds accel ~ mu times !r Zeds PREDICT Zeds 1 2 3 4 5 6 7 8 9 10 11 12 !PAC !VPV</pre> where <code>times</code> is a covariate, <code>MyZeds.txt</code> has a leading key variable listing the unique times followed by 12 base variables. With <code>!PAC</code> also specified, the (default) predict points 1:12 (in the <code>PREDICT</code> line) index the base variables rather than being covariate values for <code>times</code> . Job output qualifier <code>!CINV Zeds</code> is another way of getting the variance matrix for the 12 effects. |
| <code>!USE <i>t</i></code>                  | causes <b>ASReml</b> to set up a prediction model based on the default rules and then adds the terms listed in <i>t</i> .  |
| <b>printing</b>                             |  |
| <code>!DEC [<i>n</i>]</code>                | gives the user control of the number of decimal places reported in the table of predicted values where <i>n</i> is 0...9. The default is 4. <code>G15.9</code> format is used if <i>n</i> exceeds 9.<br><br>When <code>!VVP</code> or <code>!SED</code> are used, the values are displayed with 6 significant digits unless <i>n</i> is specified and even; then the values are displayed with 9 significant digits.   |
| <code>!ESTIMATE [<i>coef</i>]</code>        | instructs <b>ASReml</b> to form the full variance/covariance matrix for the predicted values and then calculate a series of linear functions across the set of predictions. For example, if we have a factor <code>SNP</code> with 3 classes AA, AB, BB in that order,<br><pre>PREDICT SNP !ESTIMATE -1 0 1 -1 2 -1</pre> would report 2 contrasts over the 3 predicted values. See Section 10.3.6 for additional examples.  |
| <code>!PLOT [<i>x</i>]</code>               | instructs <b>ASReml</b> to attempt a plot of the predicted values. This qualifier is only applicable in versions of <b>ASReml</b> linked with the <b>WINTERACTER Graphics</b> library. If there is no argument, <b>ASReml</b> produces a figure of the predicted values as best it can. The user can modify the appearance by pressing <code>ESC</code> to popup a menu or with the plot arguments listed in Table 10.3.   |



## 10.3 Prediction

| qualifier                     | action  |
|-------------------------------|---|
| <code>!PRINTALL</code>        | instructs <b>ASReml</b> to print the predicted value, even if it is not of an estimable function. By default, <b>ASReml</b> only prints predictions that are of estimable functions.  |
| <code>!SED</code>             | requests all <i>standard errors of difference</i> 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 <code>!SED</code> is specified, the average of the individual SED values is reported.  |
| <code>!SPLIT</code>           | on any predict statement causes the predict output from each prediction to be split into multiple files for easier parsing. The default <code>.pvs</code> file holds all explanatory text with the prediction tables. With <code>!SPLIT</code> , the predicted value lines are reported in an <code>basename_PVT_ii.txt (.csv)</code> file in a form easily loaded into <b>Excel</b> or <b>R</b> . All explanatory text associated with the predict table are written to <code>basename_PVH_ii.txt</code> file where <code>ii</code> is the index of the <code>PREDICT</code> statement.  |
| <code>!TDIFF</code>           | requests <i>t</i> -statistics be printed for all combinations of predicted values. It creates a pairwise table of <i>t</i> -statistics among all predicted values (assuming the number of cells predicted is not ridiculously large). <b>ASReml</b> reports Bonferroni and Tukey critical values for assessing the significance of the <i>t</i> -statistics in some common situations. For a 1-way table, the denominator degrees of freedom for the factor needs to have been calculated for the Wald F-table. For a 2-way table, the variance model needs to be simple, such as a split-plot, in which stratum variances as well as denominator degrees of freedom are available.                           |
| <code>!TURNINGPOINTS n</code> | requests <b>ASReml</b> 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 <code>\$TPn</code> . This was added to facilitate location of putative QTL (Gilmour, 2007).   |
| <code>!TWOSTAGEWEIGHTS</code> | 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 <code>.pvs</code> file. These values are used in some variety testing programs in Australia for a subsequent second stage analysis across many trials (Smith <i>et al.</i> , 2001). A database 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 database, are used as weights in the combined analysis. |
| <code>!VPV</code>             | requests that the variance matrix of predicted values be printed to the <code>.pvs</code> file.   |

### `!PLOT` **graphic control qualifier**

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 pressing `ESC` when the figure is displayed.

The !PLOT qualifier has the following options presented in Table 10.3.

Table 10.3: List of predict plot options

| option   | action   |
|--|--|
| <b>lines and data</b>  |  |
| <code>^addData</code>  | 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 ( <i>e.g.</i> all data points in the same stratum). |
| <code>^addlabels <i>factors</i></code>                         | 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.  |
| <code>^noSEs</code>  | specifies that no error bars should be plotted (by default, they are plotted).   |
| <code>^semult <i>r</i></code>                                  | specifies the multiplier of the SE used for creating error bars (default=1.0).   |
| <code>^joinmeans</code>  | specifies that the predicted values should be joined by lines (by default, they are only joined if the x-axis variable is numeric).  |
| <b>predictions involving two or more factors</b>               |  |
|  | 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.  |
| <code>^xaxis <i>factor</i></code>                              | specifies the prediction factor to be plotted on the x-axis.   |
| <code>^superimpose <i>factors</i></code>                       | specifies the prediction factors to be superimposed on the one panel.  |
| <code>^condition <i>factors</i></code>                         | specifies the conditioning factors which define the panels. These should be listed in the order that they will be used.  |
| <b>layout</b>  |  |
| <code>^goto <i>n</i></code>                                    | specifies the page to start at, for multi-page predictions.  |
| <code>^saveplot <i>filename</i></code>                         | specifies the name of the file to save the plot to.  |
| <code>^layout <i>rows cols</i></code>                          | specifies the panel layout on each page.   |
| <code>^bycols</code>   | specifies that the panels be arranged by columns (default is by rows).   |
| <code>^blankpanels <i>n</i></code>                             | specifies that each page contains <i>n</i> blank panels. This sub-option can only be used in combination with the layout sub-option.   |
| <code>^extrablanks <i>n</i> and<br/>^extraspan <i>p</i></code> | specifies that an additional <i>n</i> blank panels be used every <i>p</i> pages. These can only be used with the layout sub-option.  |
| <b>improving the graphical appearance (and readability)</b>    |  |
| <code>^labcharsize <i>n</i></code>                             | specifies the relative size of the data points/labels (default=0.4).   |
| <code>^panelcharsize <i>n</i></code>                           | specifies the relative size of the labels used for the panels (default=1.0).   |
| <code>^vertxlab</code>   | specifies that vertical annotation be used on the x-axis (default is horizontal).  |

## 10.3 Prediction

| option                          | action  |
|---------------------------------|---|
| <code>^abbrdlab <i>n</i></code> | specifies that the labels used for the data be abbreviated to <i>n</i> characters.              |
| <code>^abbrxlab <i>n</i></code> | specifies that the labels used for the x-axis annotation be abbreviated to <i>n</i> characters. |
| <code>^abbrslab <i>n</i></code> | specifies that the labels used for superimposed factors be abbreviated to <i>n</i> characters.  |

### 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 **!IGNORED**. 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  $X$  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.

### 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

## 10.3 Prediction

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 15 trials with classification by region and location as shown in Table 10.4: Trials classified by region and location and their means

|             |    | Location |            |    |        |              |          |     |     |
|-------------|----|----------|------------|----|--------|--------------|----------|-----|-----|
|             |    | L1       | L2         | L3 | L4     | L5           | L6       | L7  | L8  |
| Region      | R1 | T1, T2   | T3, T4, T5 | T6 |        |              |          |     |     |
|             | R2 |          |            |    | T7, T8 | T9, T10, T11 | T12, T13 | T14 | T15 |
| Trial Means |    | 10, 12   | 11, 12, 13 | 13 | 11, 13 | 11, 12, 13   | 10, 12   | 10  | 10  |

with its trial means.

Table 10.4: Trials classified by region and location and their means

|             |    | Location |            |    |        |              |          |     |     |
|-------------|----|----------|------------|----|--------|--------------|----------|-----|-----|
|             |    | L1       | L2         | L3 | L4     | L5           | L6       | L7  | L8  |
| Region      | R1 | T1, T2   | T3, T4, T5 | T6 |        |              |          |     |     |
|             | R2 |          |            |    | T7, T8 | T9, T10, T11 | T12, T13 | T14 | T15 |
| Trial Means |    | 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 !ASAVE 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: Trials classified by region and location and their means

|             |    | Location |            |    |        |              |          |     |     |
|-------------|----|----------|------------|----|--------|--------------|----------|-----|-----|
|             |    | L1       | L2         | L3 | L4     | L5           | L6       | L7  | L8  |
| Region      | R1 | T1, T2   | T3, T4, T5 | T6 |        |              |          |     |     |
|             | R2 |          |            |    | T7, T8 | T9, T10, T11 | T12, T13 | T14 | T15 |
| Trial Means |    | 10, 12   | 11, 12, 13 | 13 | 11, 13 | 11, 12, 13   | 10, 12   | 10  | 10  |

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 !ASAVE 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.

## 10.3 Prediction

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: Trials classified by region and location and their means

|             |    | Location |            |    |        |              |          |     |     |
|-------------|----|----------|------------|----|--------|--------------|----------|-----|-----|
|             |    | L1       | L2         | L3 | L4     | L5           | L6       | L7  | L8  |
| Region      | R1 | T1, T2   | T3, T4, T5 | T6 |        |              |          |     |     |
|             | R2 |          |            |    | T7, T8 | T9, T10, T11 | T12, T13 | T14 | T15 |
| Trial Means |    | 10, 12   | 11, 12, 13 | 13 | 11, 13 | 11, 12, 13   | 10, 12   | 10  | 10  |

, 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 hyper-table. If the lowest stratum is random, it may be appropriate to ignore it.

Omitting it from the !ASSOCIATE list will allow it to re-enter 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 analogous `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

```
PREDICT 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
```

## 10.3 Prediction

```
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 00,
        36 0 0 53 23 24 54 54 43 35 00,
        70 0 21 17 0 0 0 70 0 0 530,
        53 56 22 92 19 44 0 0 36 0 049,
        0 220 53 70 22 0 51 16 51 00}/5
PREDICT crop1 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 Estimate linear functions of predicted values

An !ESTIMATE *coeff* qualifier is available for the PREDICT statement. ASReml uses *coeff* to calculate a linear function of the predicted means. *coeff* may contain several sets of coefficients. The coefficients may define a formal contrast among the fitted means (coefficients sum to zero). Multiple sets of coefficients are fitted independently of each other across the set of predicted values. For example, in the oats example above, the fitted nitrogen means are: 79.3889, 98.8889, 114.2222, and 123.3889. Applying the -3, -1, 1, 3 contrast to the means gives 147.333 (corresponding to a regression coefficient of 7.3666).

```
PREDICT Nitrogen !ESTIMATE -3 -1 1 3 -1 1 1 -1 -1 3 -3 1
```

reports

| Function | Stnd_Error | t-value | Coefficients |
|----------|------------|---------|--------------|
|----------|------------|---------|--------------|



## 10.3 Prediction

```
-147.3333    14.0271   -10.50    -3.000    -1.000     1.000     3.000
 10.3333     6.2731     1.65    -1.000     1.000     1.000    -1.000
  2.0000    14.0271     0.14    -1.000     3.000    -3.000     1.000
```

The  $t$ -value for the first contrast (10.50) matches the  $F$  ( $110.32 = t^2$ ) from the Wald  $F$  table for `LinN` defined as

```
!DEFINE LinN Nitrogen -3 -1 1 3
```

but the reported regression coefficient for `LinN` from the model is not 147.3333.

In another example,

```
PREDICT SNP !ESTIMATE -1 0 1 -1 2 -1
```

would reports two linear functions over the three predicted values:

```
SNP      Predicted_Value Standard_Error Ecode
AA              0.6727         0.2769 E
AB              0.1099         0.0304 E
BB             -0.0544         0.0206 E
```

```
Estimate Stnd_Error t-value Coefficients
-0.7271   0.2777   -2.62   -1.000    0.000    1.000
-0.3984   0.2843   -1.40   -1.000    2.000   -1.000
```

**ASReml** will report if the functions estimated are not strictly contrasts (*i.e.* sum to zero). Functions may not be *Estimable* if all the predicted values are not *Estimable*. Functions are calculated independently.

### 10.3.7 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.

### 10.3.8 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**, *i.e.*

$$b_i = \sum_{j=1}^c bc_{ij} w_j.$$

The directive `TPREDICT` allows this to be produced more computationally efficiently than it would be using `PREDICT`. It is of the form

```
TPREDICT C !AVERAGE B weights !ONLYUSE fun(B).fun(C)
```

For example

```
TPREDICT Animal !AVERAGE 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\ASReml43\bin\Asreml nin89.as
```

will run the file `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

## 11.3 Command line options

---

```
ASReml -w8 rat.as
```

tells ASReml to run the job `rat.as` with workspace allocation of 8Gb.

- *arguments* provide a mechanism (mostly for regular 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, (instead of a `.as` 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 `.csv`, `.dat` or `.txt` is specified, ASReml assumes the data file has field labels in the first row and generates a `.as` file template.

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

## 11.3 Command line options

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

```
-LNW16
```

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
```

**Error! Reference source not found.** 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.

### 11.3.1 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
```

Table 11.1: Command line options

| option                                      | qualifier | type          | action  |
|---|-----------|---------------|---|
| <b>frequently used command line options</b> |           |               |   |
| 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 <i>basename.as1</i> file.                             |
| N   | !NOGRAPHS | graphics      | suppress interactive graphics.  |

## 11.3 Command line options

| option                            | qualifier                  | type            | action  |
|-----------------------------------|----------------------------|-----------------|---|
| <i>Ww</i>                         | <b>!WORKSPACE</b> <i>w</i> | workspace       | set workspace size to <i>w</i> Mbyte.   |
| <b>other command line options</b> |                            |                 |   |
|                                   | <b>!ARGS</b> <i>a</i>      | job control     | to set arguments ( <i>a</i> ) in job rather than on command line.   |
| <i>A</i>                          | <b>!ASK</b>                | job control     | prompt for options and arguments.   |
| <i>Bb</i>                         | <b>!BRIEF</b> <i>b</i>     | output control  | reduce output to <i>.asr</i> file.  |
| <i>D</i>                          | <b>!DEBUG</b>              | debug           | invoke debug mode.  |
| <i>E</i>                          | <b>!DEBUG</b> 2            | debug           | invoke extended debug mode.   |
| <i>Gg</i>                         | <b>!GRAPHICS</b> <i>g</i>  | graphics        | set interactive graphics device.  |
| <i>Hg</i>                         | <b>!HARDCOPY</b> <i>g</i>  | graphics        | set interactive graphics device, graphics screens not displayed.  |
| <i>I</i>                          | <b>!INTERACTIVE</b>        | graphics        | display graphics screen.  |
| <i>O</i>                          | <b>!ONERUN</b>             | job control     | override rerunning requested by <b>!RENAME</b> .  |
|                                   | <b>!OUTFOLDER</b>          | output control  | changes output folder.  |
| <i>P</i>                          | <b>NA</b>                  | post-processing | calculation of functions of variance components.  |
| <i>Q</i>                          | <b>!QUIET</b>              | graphics        | suppress screen output.   |
| <i>Rr</i>                         | <b>!RENAME</b>             | job control     | repeat run for each argument renaming output filenames.   |
| <i>Yv</i>                         | <b>!YVAR</b> <i>v</i>      | job control     | over-ride <i>y</i> -variate specified in the command file with variate number <i>v</i> .  |
| <i>Z</i>                          | <b>NA</b>                  | license         | reports current license details.  |
| <i>X</i>                          | <b>!XML</b>                | output control  | requests that the main output from the <i>.asr</i> , <i>.pvs</i> and <i>.sln</i> files be also written in the <i>.xml</i> file. |

### 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 always written to the working 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.

### 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 except to send to the link <https://vsni.co.uk/support> when a job crashes for further evaluation. 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 qualifiers are given in Table 11.2, file content substrings in Table 11.3 and File type qualifiers in Table 11.4.

Table 11.2: Graphic display options

| option  | action   |
|---|--|
| <b>I</b> ( <b>!INTERACTIVE</b> )                    | The option permits the variogram and residual graphics to be displayed. This is the default unless the <b>L</b> option is specified.   |
| <b>N</b> ( <b>!NOGRAPHICS</b> )                     | The option prevents any graphics from being displayed. This is the default when the <b>L</b> option is specified.  |
| <b>G</b> [ <i>g</i> ] ( <b>!GRAPHICS</b> <i>g</i> ) | The option sets the file type for hard copy versions of the graphics. Hard copy is formed for all the graphics that are displayed.   |
| <b>H</b> [ <i>g</i> ] ( <b>!HARDCOPY</b> <i>g</i> ) | This option replaces the <b>G</b> option when graphics are to be written to file but not displayed on the screen. The <b>H</b> may be followed by a format code <i>e.g.</i> <b>H22</b> for <code>.eps</code> . |
| <b>Q</b> ( <b>!QUIET</b> )                          | This option is used when running under the control of <b>ASReml-W</b> to suppress any <b>POP-UPS/PAUSES</b> from <b>ASReml</b> .   |

## 11.3 Command line options

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 Table 11.3).
- `<pass>` is inserted when the job is repeated (!RENAME or !CYCLE) to ensure filenames are unique across repeats.
- `<section>` is inserted to distinguish files produced from different sections of data (for example from multisite spatial analysis).
- `<ext>` indicates the file graphics format.

Table 11.3: Graphic file content substrings

| <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>i</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

Table 11.4: Graphic file type qualifiers

| <i>g</i> | 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   |



## 11.3 Command line options

| g  | qualifier | description             | <ext> |
|----|-----------|-------------------------|-------|
| 12 | !HPGL 2   | HP-GL2                  | hgl   |
| 21 | !PNG      | PNG                     | png   |
| 22 | !EPS      | Encapsulated PostScript | eps   |

### 11.3.5 Job control command line options (C, F, O, R, Y)

ASReml has a few job control commands that are useful when running modes in batch. The list of these are presented in [Table 11.5](#).

Table 11.5: Job control command options

| option         | qualifier             | Action   |
|----------------|-----------------------|--|
| C              | !CONTINUE             | indicates that the job is to continue iterating from the values in the <code>.rsv</code> file. This is equivalent to setting <code>!CONTINUE</code> on the datafile line, see <a href="#">Table 5.7</a> for details.   |
| F)             | !FINAL                | indicates that the job is to continue for one more iteration from the values in the <code>.rsv</code> file. This is useful when using <code>PREDICT</code> , see <a href="#">Chapter 10</a> .  |
| O              | !ONERUN               | is used with the <code>R</code> option to make <b>ASReml</b> perform a single analysis when the <code>R</code> option would otherwise attempt multiple analyses. The <code>R</code> option then builds some arguments into the output file name while other arguments are not. For example,<br><code>ASReml -nor2 mabphen 2 TWT out(621) out(929)</code><br>results in one run with output files <code>mabphen2_TWT.*</code> .   |
| R [ <i>r</i> ] | !RENAME [ <i>r</i> )] | is used in conjunction with at least <i>r</i> argument(s) and does two things: it modifies the output filename to include the first <i>r</i> arguments so the output is identified by these arguments, and, if there are more than <i>r</i> arguments, the job is rerun moving the extra arguments up to position <i>r</i> (unless <code>!ONERUN (O)</code> is also set). If <i>r</i> is not specified, it is taken as 1. For example,<br><code>ASReml -r2 job wwt gfw fd fat</code><br>is equivalent to running three jobs:<br><code>ASReml -r2 job wwt gfw → jobwwt_gfw.asr</code><br><code>ASReml -r2 job wwt fd → jobwwt_fd.asr</code><br><code>ASReml -r2 job wwt fat → jobwwt_fat.asr</code> |
| Yy             | !YVAR                 | overrides the value of <i>response</i> , the variate to be analysed (see <a href="#">Section 6.2</a> ) with the value <i>y</i> , where <i>y</i> is the <i>number</i> of the data field containing the trait to be analysed. This facilitates analysis of several traits under the same model. The value of <i>y</i> is appended to the <i>basename</i> so that output files are not overwritten when the next trait is analysed.   |

### 11.3.6 Workspace command line options (W)

We have reworked some of the core routines to allow access to a larger workspace (up from 32 to 500 Gbyte workspace). The use of memory has been changed to separate different kinds of data into different arrays in accord with more modern programming conventions. This results in the need to allocate more memory for a particular job than was needed in **ASReml 4.1**.

Workspace can be set either on the command line (with the `-Wm` option) or with the `!WORKSPACE m` qualifier on the first line of the job (above the `TITLE` line). The former takes precedence. The argument `m` can include a decimal point and should be between 0.2 and 500; `m` greater than 500 is treated as Mbytes. **ASReml** reports the space available in Gbytes to one decimal place and a value of `m` less than 0.2 is interpreted as 0.2 Gbytes.

The default workspace in **ASReml 4.2** is 2 Gbytes, which is ample for the majority of runs. The minimum allocated is 0.6 Gbytes; the maximum allocated depends on what is available on your PC. We recommend that the workspace requested not exceed the RAM available on your machine (commonly 8 or 16 Gbytes). 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 be unhappy. **ASReml** reports the actual amount of primary workspace used in a job at the end of the `.asr` file. Sometimes the allocation of primary workspace means that **ASReml** cannot access sufficient secondary workspace. **ASReml** will report this and suggest the primary workspace be reduced.

### 11.3.7 Examples Command Line

Table 11.6: Job control command examples

| ASReml code                           | action   |
|---------------------------------------|--|
| <code>asrem1 -LW64 rat.as</code>      | set workspace to 64 Mbyte, send screen output to <code>rat.asl</code> and suppress interactive graphics.   |
| <code>asrem1 -IL rat.as</code>        | send screen output to <code>rat.asl</code> but display interactive graphics.   |
| <code>asrem1 -N rat.as</code>         | allow screen output but suppress interactive graphics.   |
| <code>asrem1 -ILW512 rat.as</code>    | increase workspace to 512 Mbyte, send screen output to <code>rat.asl</code> but display interactive graphics.  |
| <code>asrem1 -rw1 coop wwt ywt</code> | runs <code>coop.as</code> twice using 1Gbyte workspace and writing results to <code>coopwwt.ext</code> and <code>coopyywt.ext</code> respectively and substituting <code>wwt</code> and <code>ywt</code> for <code>\$1</code> in the two runs, respectively. |

## 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 16.13.

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

## 11.4 Advanced processing arguments

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 11.7: The use of arguments in **ASReml**

| in command file                 | on command line   | becomes in <b>ASReml</b> 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<br>(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 complete 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 high-level qualifiers `!CYCLE` and `!DOPATH` enable multiple analyses to be defined and run in one execution of **ASReml**.

Table 11.8: High-level qualifiers

| qualifier                                 | action  |
|---|---|
| <code>!ASSIGN <i>list</i></code>          | <p>an <code>!ASSIGN <i>string</i></code> 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 <code>!ASSIGN</code> must start on its own input line. The syntax is</p> <pre>!ASSIGN <i>name string</i></pre> <p>or</p> <pre>!ASSIGN <i>name</i> !&lt; <i>string</i> !&gt;</pre> <p>and the defined <i>string</i> is substituted into the job where <i>\$name</i> appears. <i>string</i> is the rest of the line and may include blanks. If <code>!&lt; !&gt;</code> encloses <i>string</i>, <i>string</i> may extend over several lines, which are concatenated. For example</p> <pre>!ASSIGN TVS xfa1(Treat) ... ... \$TVS.geno ...</pre> <p>is interpreted as</p> <pre>... xfa1(Treat).geno ...</pre> <p><b>Restrictions</b></p> <ul style="list-style-type: none"> <li>• A maximum of 50 assign strings may be defined.</li> <li>• The combined length of all strings is 5000 characters.</li> <li>• <i>name</i> may have up to 8 characters but should not begin with a number (see command line arguments).</li> <li>• Dollar substitution occurs before most other high-level actions. ASSIGN strings and command line arguments may substitute into a <code>!CYCLE</code> line.</li> <li>• <code>I</code>, <code>J</code>, <code>K</code> and <code>L</code> are reserved as names referring to items in the <code>!CYCLE</code> list and should therefore not be used as names of an ASSIGN string.</li> </ul>  |
| <pre>!CYCLE [!SAMEDATA] <i>list</i></pre> | <p>is a mechanism whereby <b>ASReml</b> can loop through a series of jobs. The <code>!CYCLE</code> has a qualifier <code>!SAMEDATA</code> that tells <b>ASReml</b> to use the same data for all cycles, <i>i.e.</i> the data file is only read on the first cycle, and is kept in memory for later cycles. The <code>!CYCLE</code> qualifier must appear on its own line. <i>list</i> is a series of values which are substituted into the job wherever the <code>\$I</code> 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>i : j</i> (no embedded spaces). The output from the set of runs is concatenated into a single set of files, but the output written to the <code>.asr</code> file is slightly abbreviated after the first cycle, by suppressing the data summary and fixed effect solutions that might otherwise appear (see <code>!BRIEF</code>; the <code>!BRIEF</code> qualifier is set after the first cycle). For example,</p> <pre>!CYCLE 0.4 0.5 0.6 mat2(Xpos 1.9 \$I !GPF)</pre> <p>would result in three runs and the results would be appended to a single file.</p> <p>Putting <code>!SAMEDATA</code> on the (leading) <code>!CYCLE</code> line makes <b>ASReml</b> read the data (and <code>.grr</code> file) file in the first <code>CYCLE</code> and hold it in memory for use in subsequent cycles. This is advantageous when the data/<code>.grr</code> file is large and there are many cycles to execute where the model changes, but the data/<code>.grr</code> file doesn't.</p> <p>The <code>!CYCLE</code> mechanism acts as an inner loop when used with <code>!RENAME !ARG</code>. As an example, the <code>!RENAME !ARG</code> arguments might list a set of traits, and the <code>!CYCLE</code> arguments sequentially test a set of markers.</p> |

## 11.4 Advanced processing arguments

| qualifier                              | action   |
|--|--|
|  | <p>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,</p> <pre>!CYCLE Y1;X1 Y2;X2 \$I ~ mu \$J</pre> <p>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,</p> <pre>LogL: LogL Residual NEDF NIT Cycle Text LogL: -208.97 0.703148 587 6 1466 "LogL Converged"</pre> <p>The LogL: line with the highest LogL value is repeated at the end of the .asr file.</p>  |
| !DOPATH <i>n</i><br>!DOPART <i>n</i>   | <p>!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 interchangeably. Which particular lines in the .as file are honoured is controlled by the argument <i>n</i> of the !DOPATH qualifier in conjunction with !PATH (or !PART) statements.</p> <p>The argument (<i>n</i>) 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 16.13 for examples. The default value of <i>n</i> is 1.</p> <p>!DOPATH <i>n</i> 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.</p>  |
| !FOR <i>forlist</i> !DO <i>command</i> | <p>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. <i>list</i> may be an assign string since they are processed before the !FOR statement is expanded. Furthermore, if <i>list</i> is entirely integer numbers, <i>i:j</i> notation can be used. For example</p> <pre>!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</pre> <p>is expanded to</p> <pre>!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</pre> <p>The aim here is to generate the 3 !MBF statements required to extract markers 35, 75 and 125 from the marker file <i>markers.csv</i>. 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<i>lettersinteger</i> is interpreted as !RFIELD <i>integer</i> so the !FOR statement can be written even more concisely as</p> <pre>!FOR \$Markers !DO !MBF mbf(Geno,1) markers.csv !key 1 !RFIELD\$S !RENAME \$S</pre> <p>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.</p> <p><b>Restriction</b> <i>forlist</i> and <i>command</i> are both limited to 200 characters</p> |

## 11.4 Advanced processing arguments

| qualifier   | action  |
|---|---|
| <code>!IF <i>string1</i> == <i>string2</i> <i>text</i></code> | <p>One form of the IF statement is</p> <pre>!IF string1 == string2 !ASSIGN M1 brt DamAge</pre> <p>which makes the !ASSIGN statement active if <code>string1</code> is the same as <code>string2</code>. Note that there need to be spaces before and after <code>==</code> 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</p> <pre>: !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)</pre> |
| <code>!PATH <i>pathlist</i></code>                            | <p>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 <i>a : b</i> notation. For example,</p> <pre>mydata.asd !DOPATH 4 !PATH 2 4 6:10</pre> <p>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.</p>                        |

### Example

The following code will run through 1,000 models fitting 1,000 different marker variables to some data. For processing efficiently the 1,000 marker variables are held in 1,000 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
```

## 11.5 Software performance

---

```
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 Software performance

### 11.5.1 Timing process

Timing information useful for comparing execution time between models and/or builds of ASReml is available in the .asl file if the !LOGFILE and !DEBUG qualifiers (or command line options -DL) are set on the first line of the job (above the TITLE line). Running the command `grep '>>' job.asl` at the command prompt will extract timing information. (grep is a UNIX/Linux command-line option used to find a specific string from inside a file. For Windows, the grep alternative is findstr).

We give two examples of timings using ASReml 4.3. The first example is of the comp of the comparison of XFA and RRD models as discussed in Section 7.11.6. The second example is of trimming discussed in Section 9.9.

#### XFA/RRD Example

This is a complex multi environment analysis with 24 experiments and 884 genetic lines linked with a *G* matrix for a total of 12,354 records. The model fitted is:

```
yield ~ mu Experiment
!r Rep.Experiment Block.Rep.Experiment grm(Line).xfal(Experiment)
```

and it contains 22,797 equations. The log-file output for this analysis is

```
>> Windows x64 30.0 Gbyte PhenoG2F8.ni/PhenoG2F 17 Apr 2025 08:59:00.497
>> >> ASReml Process Clock SumClock
>> >> Getting Started: sec 1.04 1.04
>> >> Before Order : sec 0.20 1.24
```

## 11.5 Software performance

---

```
>> >>      * Ordering : sec      2.23      3.48
FILLIN 26879705 ==>> 27580197      1.03, #SR: 22772, AvLen: 1211, MxLen: 2023, AvFlops: 818711
>> >>      C reordered: sec      4.63      5.87
>> >>      Cabs blocks      407  49.93858      95
>> >>      C absorbed: sec      3.32      9.19
>> >>      WV formed: sec      0.06      9.25
>> >>      AI formed: sec      0.38      9.63
>> >>      Ci formed: sec      4.22      13.86
>> >>      373 >> Ci nodal blocks; average      55.13137
>> >>      Ci reordered: sec      1.68      15.54
>> >>      Gscore done: sec      1.20      16.75
>> >>      Iteration complete: sec      0.02      16.77
>> >>      Iteration complete: sec      13.24      30.01
FILLIN 25780785 ==>> 26263304      1.02, #SR: 22772, AvLen: 1153, MxLen: 1781, AvFlops: 772350
>> >>      Iterations done: sec      152.06      182.07
>> >>      SLN written: sec      0.12      182.19
>> >>      YHT written: sec      0.07      182.26
>> >>      Finished: sec      0.00      182.26
```

This job required 13 iterations in 182 seconds (that is, 13.2 seconds per iteration), and had a log-likelihood value of -46,808.1.

In ASReml 4.3, it is now possible to fit the following equivalent model

```
yield ~ mu Experiment,
!r Rep.Experiment Block.Rep.Experiment,
rr1(Experiment).grm(Line) diag(Experiment).grm(Line)
```

that, in contrast, required 3.3 seconds per iteration, for a total of 15 iterations with a very similar log-likelihood value of -46,810.2.

```
>> Windows x64 30.0 Gbyte PhenoG2F9/PhenoG2F 17 Apr 2025 10:28:11.087
>> >>      ASReml Process      Clock      SumClock
>> >>      Getting Started: sec      1.03      1.03
>> >>      Before Order : sec      0.07      1.10
>> >>      * Ordering : sec      1.22      2.32
FILLIN 9391895 ==>> 10424393      1.11, #SR: 43988, AvLen: 237, MxLen: 862
>> >>      C reordered: sec      1.69      2.79
>> >>      Cabs blocks      292  73.28082      95
>> >>      C absorbed: sec      0.70      3.49
>> >>      WV formed: sec      0.20      3.68
>> >>      AI formed: sec      0.12      3.81
>> >>      Ci formed: sec      0.70      4.51
>> >>      233 >> Ci nodal blocks; average      88.08154
>> >>      Ci reordered: sec      0.38      4.89
>> >>      Gscore done: sec      0.77      5.66
>> >>      Iteration complete: sec      0.01      5.67
>> >>      Iteration complete: sec      3.29      8.96
>> >>      Iterations done: sec      44.33      53.29
>> >>      SLN written: sec      0.22      53.51
>> >>      Finished: sec      0.07      53.58
```

Typically, as shown above, the major components are: Order found, C absorbed and Ci formed. The FILLIN line reports the size of the **C** matrix before and after absorption, and the ratio (after/before) of the sizes.

### Trimming Example

Consider the analysis of a breeding experiment for wheatgrass, a perennial wheat species. The analysis is of 22 harvests taken from 10 trials planted over 4 years. The selection of genotypes planted in a trial differ considerably, the later plantings being lines (genotypes) derived from earlier



plantings. A simplified model for the trait plant height, PTHT, using a genotype relationship matrix (marker based) to provide genetic links is

```
PTHT ~ mu mv !r Harvest rrl(Harvest).grml(geno) at(Harvest).grml(geno) female.male  
residual sat(Harvest).arlv(row).arl(range)
```

This corresponds to a factor analytic model where the term provides the overall genetic covariances while the `at(Harvest).grml(geno)` term provides the specific genotype by harvest variance components for each harvest.

The coefficient matrix includes 23 instances of the full genomic matrix which is of order 4,482. An alternative way of setting up the model is just to include the genotypes with data in the **GRM** used for estimating the 22 specific variance components. The matrices vary in size from 308 to 576 genotypes. The model in this case is

```
PTHT ~ mu mv !r Harvest rrl(Harvest).grml(geno) sat(Harvest).grml(geno) female.male  
residual sat(Harvest).arlv(row).arl(range)
```

The variance components from this model are the same as for the full model, but this model took 33.3 seconds per iteration compared with 116.7 seconds per iteration for the original model. Comparison of the model processing details are presented in the table below.

Table 11.9: Model processing details comparison

| action               | using at () | using sat () |
|----------------------|-------------|--------------|
| Initialize (sec)     | 1           | 1            |
| Ordering sec (sec)   | 50          | 7            |
| Absorption sec (sec) | 43          | 4            |
| Forming AI sec (sec) | 13          | 13           |
| Sparse Inverse(sec)  | 40          | 4            |
| Score (sec)          | 16          | 13           |
| Equations            | 367,587     | 278,397      |
| C matrix cells       | 214,560,191 | 12,453,485   |

### 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 takes 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`).

## 11.5 Software performance

---

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`).

# 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 accommodate various scenarios. It was developed with assistance from Chandrapal Kailasanathan to replace the `!MERGE` qualifier (see **Error! Bookmark not defined.**) 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

```
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  |
|------------------------------------|---|
| <code>!CHECK</code>                | requests <b>ASReml</b> confirm that fields having a common name have the same contents. Discrepancies are reported to the <code>.asr</code> file. If there are fields with common names which are not <i>key</i> fields, and <code>!CHECK</code> is omitted, the fields will be assumed different and both versions will be copied.   |
| <code>!KEY <i>keyfields</i></code> | 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 <code>!KEY</code> qualifier may be omitted altogether. If key fields are not nominated and there are no common field names, the files are interleaved.   |
| <code>!KEEP</code>                 | instructs <b>ASReml</b> 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. <code>!KEEP</code> may be specified with either or both input files.  |
| <code>!NODUP <i>fields</i></code>  | 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 <code>!NODUP</code> 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 <code>!NODUP bwt wwt</code> will copy these traits only once into the merged file. |
| <code>!SKIP <i>fields</i></code>   | is used to exclude fields from the merged file. It may be specified with either or both input files.  |
| <code>!SORT</code>                 | instructs <b>ASReml</b> 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.

# 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 !ridv(Sire)
residual idv(units)
VPREDICT !DEFINE
F phenvar idv(Sire) +
idv(units)
F genvar idv(Sire)*4
```

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 C, D, F, H, K, M, R, S, V, X or W) must occur in column 1
  - C is a copy function to facilitate rearranging variance components
  - D is a delete function
  - F forms linear combinations of variance components
  - H is for forming heritabilities, the ratio of two components,
  - K sets a vector (or matrix) of coefficients for use by M
  - M pre/post multiplies a US matrix by the K matrix
  - R is for forming the correlation from a covariance component
  - S is a square root function

## 13.2 Syntax

---

- V is for converting components related to a **CORUH** or an **XFA** structure into components related to a **US** structure
- W is an output (write) function
- 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) arlv(B) nrm(C).us(Trait) D
residual id(units).us(Trait)
```

The covariance functions with parameters

```
idv(A)
arlv(B)
us(Trait)
```

in

```
nrm(C).us(Trait)
```

and

```
us(Trait)
```

in

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

are named

```
idv(A)
arlv(B);arlv(B)
nrm(C).us(Trait);us(Trait)
```

and

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

respectively.

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)
arlv(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 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

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  $k + \mathbf{c}'\mathbf{v}$  and forms  $\text{cov}(\mathbf{c}'\mathbf{v}, \mathbf{v})$  and  $\text{var}(\mathbf{c}'\mathbf{v})$  where  $\mathbf{v}$  is the vector of existing variance components,  $\mathbf{c}$  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 * cb + 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  $\mathbf{v}$  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

```
y ~ mu !r idv(Sire)
residual idv(units)
VPREDICT !DEFINE
F phenvar idv(Sire) +
idv(units)
F genvar idv(Sire)*4
. . .
```

## 13.2 Syntax

```
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  $\sigma_n^2/\sigma_d^2$  and  $se[\sigma_n^2/\sigma_d^2]$  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
```

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  $\mathbf{v}$  with variance matrix  $\mathbf{V}$ , if

$$v_a = f_a(\mathbf{v})$$

and

$$v_b = f_b(\mathbf{v})$$

then if

$$\delta \mathbf{v}_a = \frac{\delta f_a(\mathbf{v})}{\delta \mathbf{v}}$$

and if



$$\delta \mathbf{v}_b = \frac{\delta f_b(v)}{\delta v}$$

then

$$\text{cov}(\mathbf{v}_a, \mathbf{v}_b) = \delta \mathbf{v}_a' \mathbf{V} \delta \mathbf{v}_b$$

### 13.2.2 Convert CORUH and XFA to US

- $\forall \text{ label } i:j$  where  $i:j$  spans a CORUH variance structure, inserts the US matrix based on the CORUH parameters.
- $\forall \text{ label } i:j$  where  $i:j$  spans an XFA variance structure, inserts the US matrix based on the XFA parameters.

### 13.2.3 Correlation

Correlations are requested by lines beginning with an R. The specific form of the directive is

- $R \text{ label } a \text{ } ab \text{ } b$  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

```
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)
```

- $R \text{ label } a:n$  calculates the correlation  $r = \sigma_{ab} / \sqrt{\sigma_a^2 \sigma_b^2}$  for all correlations in the lower triangular row-wise.

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 Convert variance matrix of variable to variance matrix of transformed variable

Functions `K` and `M` facilitate calculation of transformed variance components. A model may generate an estimated variance-covariance matrix  $\Sigma$  for  $s$  effects  $\mathbf{u}$  and there might be interest in using  $\mathbf{K}$  a  $(r \times s)$  matrix to transform  $\mathbf{u}$  to  $\mathbf{Ku}$  with derived symmetric  $(r \times r)$  variance matrix  $\mathbf{M} = \mathbf{K} \Sigma \mathbf{K}'$ . For instance, we might have a yield on a plot, one of its row neighbours and one of its column neighbours estimated using a spatial autoregressive model and wish to compute the variance matrix of sums and differences of these three yields. If  $\Sigma$  is the residual variance of the three plots and  $\mathbf{K}$  a matrix allowing the formation of sums or differences then we wish to form  $\mathbf{M} = \mathbf{K} \Sigma \mathbf{K}'$ .

- `K label v` defines a vector ( $v$ ) of coefficients.
- `M label i:j m:n !DIAG` specifies  $\mathbf{K}$  an  $(r \times c)$  matrix derived from the set of  $r$  vectors of length  $c$  starting with *firstveclabel* and ending with *lastveclabel*. If all the required vectors start with the same initial symbols *initialvecsymbols* can be used to specify  $\mathbf{K}$ .
- `m:n` specifies  $\mathbf{S}$  an unstructured symmetric  $(s \times s)$  variance matrix. Note that `m:n` can be replaced by a structure label.

There are two cases if  $c = s$  the matrices  $\mathbf{K}$  and  $\mathbf{S}$  are conformable and  $\mathbf{M} = \mathbf{KSK}'$  is computed. If the columns of  $\mathbf{S}$  are a multiple ( $s/c$ ) of the columns of  $\mathbf{K}$  then a partitioned form of a  $rs/c \times rs/c$  variance matrix  $\mathbf{M}$  is formed with  $(r \times r)$  sub-matrix  $\mathbf{M}_{ij} = \mathbf{KS}_{ij}\mathbf{K}'$  with  $\mathbf{S}$  partitioned into  $(c \times c)$  sub-matrices  $\mathbf{S}_{ij}$  ( $i, j = 1, \dots, s/c$ ). If `!DIAG` is set only the diagonal elements of  $\mathbf{M}$  are computed. See Section 13.2.7 for more details.

### 13.2.5 Write components

- `W filename i[:j]` writes component(s)  $i$  [to  $j$ ] to a file (*filename.vpc*) and their variance matrix to another file (*filename.vpv*).

### 13.2.6 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) !f Trait.grp
residual id(units).us(Trait)

VPREDICT !DEFINE
F phenvar id(units).us(Trait);us(Trait) + us(Trait).id(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

## 13.2 Syntax

| 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            | 0.344370E-01 | 0.344370E-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 **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  $5/\sqrt{4 \times 6}$ , 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      0.344370E-01    0.169640E-01
F phenvar id(units).us(Trait);us(Trait) + us(Trait).id(sire);us(Trait)
  7 phenvar                          24.664      0.64250
  8 phenvar                          2.6343      0.14763
  9 phenvar                          1.6974      0.52366E-01
F addvar sire;us(Trait) * 4
 10 addvar                          5.8328      1.5926
 11 addvar                          0.52112     0.27171
 12 addvar                          0.13775     0.67806E-01
H heritA addvar[1] phenvar[1]
  heritA = addvar 10/phenvar 7=      0.2365     0.0612
H heritB addvar[3] phenvar[3]
  heritB = addvar 12/phenvar 9=      0.0812     0.0394
R phencorr phenvar
  phenco 2 1 = phenv 8/SQR[phenv 7*phenv 9]= 0.4071     0.0183
R gencorr addvar
  gencor 2 1 = addva 11/SQR[addva 10*addva 12]= 0.5814     0.2039
Note: The parameter estimates are followed by
      their approximate standard errors.
```

The first 8 lines are based on the `.asr` file.

### 13.2.7 Conversion of variance matrix variables to variance matrix of transformed variables

For instance, we might have a plot, one of its row neighbours and one of its column neighbours estimated using a spatial autoregressive model and wish to compute the variance matrix of sums and differences of these three plots. If  $\sigma^2$ ,  $\rho_r$  and  $\rho_c$  are the residual variance, autoregressive row correlation and autoregressive column parameter respectively from fitting the autoregressive model `ar1v(row).ar1(column)` Then the residual matrix of the three plots is

$$\Sigma = \begin{bmatrix} \sigma^2 & \rho_r \sigma^2 & \rho_c \sigma^2 \\ \rho_r \sigma^2 & \sigma^2 & \rho_r \rho_c \sigma^2 \\ \rho_c \sigma^2 & \rho_r \rho_c \sigma^2 & \sigma^2 \end{bmatrix}$$

The ASReml code

```
X COV12 ar1v(row).ar1(column);Residual * ar1v(row).ar1(column);ar1v(row)
X COV13 ar1v(row).ar1(column);Residual * ar1v(row).ar1(column);ar1(column)
X COV23 COV12*ar1v(row).ar1(column);ar1(column)
F US ar1v(row).ar1(column);Residual
F US COV12
F US ar1v(row).ar1(column);Residual
F US COV13
F US COV23
F US ar1v(row).ar1(column);Residual
```

Constructs the covariance terms in  $\Sigma$  (COV12, COV13 and COV23) and forms the lower triangle part of  $\Sigma$  (US).

The ASReml code

```
K KS_1 1 1 0
K KS_2 1 0 1
K KS_3 0 1 1
```

specifies 3 row vectors.

The ASReml code

```
M m_11 KS_1 US
M M_SS KS_1:KS_3 US
```

Forms the scalar  $\mathbf{m}_{11} = \mathbf{k}_{s1} \Sigma \mathbf{k}'_{s1} (\mathbf{m}_{11})$  and matrix  $\mathbf{M}_{ss} = \mathbf{K}_s \Sigma \mathbf{K}'_s (\mathbf{M}_{ss})$  with

$$\mathbf{k}_s = [1 \ 1 \ 0] \text{ and } \mathbf{K}_s = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix}$$

As all the vectors start with KS the last matrix can also be formed by replacing KS\_1:KS\_3 by KS.

Another example of requiring variance matrices of transformed variables, indeed the motivating example is random regression models. A random regression model will typically contain a term like `us(leg(dim,2)).id(individual)` which estimates a  $3 \times 3$  variance matrix, say  $\Sigma^*$ , with 6 variance parameters being (co)variances among the 3 Legendre polynomial effects, say  $\mathbf{u}$ , for individuals. We sometimes would like the variance matrix of predictions of the individual effects at specific values of dim, that is, for  $\mathbf{K}^* \mathbf{u}$  with  $\mathbf{K}^*$  containing vectors of Legendre polynomials at various values of dim available from the .res file. For example

```
K Leg_15 0.7071 -1.1431 1.2754
K Leg_45 0.7071 -0.8981 0.4848
K Leg_75 0.7071 -0.6532 -0.1159
K Leg_105 0.7071 -0.4082 -0.5270
```

Forms vectors of 3 Legendre polynomial values for dim = 15, 45, 75, 105 for polynomials with 305 points. The polynomial values are available in the .res file. Then

```
M m^*_11 leg_15 us(leg(dim,3)).id(individual);us(leg(dim,2))
```

and

```
M M^* leg us(leg(dim,3)).id(individual);us(leg(dim,2))
```

form the scalar  $\mathbf{m}_{11}^* = \mathbf{k}_1^* \Sigma \mathbf{k}_1^{*'} (\mathbf{m}_{11}^*)$  and matrix  $\mathbf{M}^* = \mathbf{K}^* \Sigma \mathbf{K}^{*'} (\mathbf{M}^*)$  with

$$\mathbf{k}_1^* = [0.7071 \ -1.1431 \ 1.2754]$$

and

$$\mathbf{K}^* = \begin{bmatrix} 0.7071 & -1.1431 & 1.2754 \\ 0.7071 & -0.8981 & 0.4848 \\ 0.7071 & -0.6532 & -0.1159 \\ 0.7071 & -0.4082 & -0.5270 \end{bmatrix}$$

The line

```
M D^* leg us(leg(dim,3)).id(individual);us(leg(dim,2)) !DIAG
```

calculates the 4 diagonal elements of  $\mathbf{M}^*$ .

The example can be extended to deal with unstructured models generated by interactions of factors with variables, for instance `us(Trait.leg(dim,2)).id(individual)` then if the factor Trait has 2 values, then `us(Trait.leg(dim,2))` generates a (6 x 6) matrix. If we were interested in calculating the variances and covariances between effects at the 4 values of dim we could use

```
M Gen2Trait leg us(Trait.leg(dim,2)).id(individual);us(Trait.leg(dim,2))
```

This line generates an (8 x 8) matrix  $\mathbf{M}$  labelled Gen2Trait containing variance and covariance at 4 points for two traits formed from

$\mathbf{K}^*$  the (4 x 3) matrix specified by Leg and

$\mathbf{S}$  a (6 x 6) matrix formed from `us(Trait.leg(dim,2))`

The first 4 rows and columns of  $\mathbf{M}$  contain variances and covariances associated with the first trait in a (4 x 4) sub-matrix  $\mathbf{M}_{11}$  constructed as  $\mathbf{K}^* \mathbf{S}_{11} \mathbf{K}^{*'} where  $\mathbf{S}_{11}$  is a (3 x 3) sub-matrix formed from the first 3 rows and columns of  $\mathbf{S}$  and contains variance and covariance parameters associated with the first trait.$

More generally the (m, n) element of  $\mathbf{M}_{ij}$  is given by  $\mathbf{k}_m^* \mathbf{S}_{ij} \mathbf{k}_n^{*'} with  $\mathbf{k}_m^*$  the m-th row of  $\mathbf{K}^*$ . These 4 indices (i, j, m, n) are used to identify individual elements in the .pvc file and we note that elements with the same indices i and m (j and n) are in the same row(column) of  $\mathbf{M}$ .$

## 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.

## 13.4 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.
```

# 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  |
|---------------------------|--|
| <b>key output files</b>   |  |
| .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 5.7.  |
| .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 5.7.   |
| .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 5.7   |
| .yht                      | contains the predicted values, residuals and diagonal elements of the hat matrix for each data point.  |
| <b>other output files</b> |  |
| .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.   |
| .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 5.5.  |
| .ass                      | contains the data summary created by the !SUM qualifier (see Table 5.6).   |
| .srs                      | contains the data and residuals in a binary form for further analysis (see !RESIDUALS, Table 5.8).   |
| .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.   |



## 14.2 An example

| file         | description  |
|--------------|--|
| .vll         | holds factor level names when data/residuals are saved in binary form. This !SAVE qualifier is defined in Table 5.8.   |
| .vpc<br>.vpv | <i>filename.vpc</i> and <i>filename.vpv</i> contains estimates of functions of variance parameters and their variance matrix respectively. The W directive in VPREDICT (see VPREDICT in <a href="#">Chapter 13</a> ) is used to specify filename and required functions. |
| .vrb         | contains the estimates of the fixed effects and their variance approximate prediction variance matrix corresponding to the dense portion of the 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 <a href="#">Chapter 13</a> ).  |
| .was         | <i>basename.was</i> 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.  |
| .wvr         | contains the working variables if !WRV set.  |
| .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** in Section 7.5 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
nin89aug.asd !SKIP 1 !DISPLAY 15
tabulate yield ~ variety
yield ~ mu variety !fmv
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 log-likelihood 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 freedom, 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 Section 2.4.1). 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 (!MAXIT is defined in Table 5.6) has been reached or successive LogL values are less than  $0.002i$  apart.

The following is a copy of nin89a.asr.

```
ASReml 4.3ni [14 Nov 2025] NIN Alliance Trial 1989 version & title
Windows x64      2.0 Gbyte workspace nin89AR 06 Dec 2025 13:56:06.377 date
* Licensed to: valid
* Your ASReml license expires in 31 days *
*****
* Contact support@asreml.co.uk for licensing and support *
***** ARG *
```

## 14.3 Key output files

```
Folder: C:\Users\...\NIN
variety !A
QUALIFIERS: !SKIP 1 !DISPLAY 15
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 | StdDevn |
|-----------------|---------|-------|-------|---------|---------|---------|---------|
| 1 variety       | 56      | 0     | 0     | 1       | 26.4545 | 56      |         |
| 2 id            |         | 0     | 0     | 1.000   | 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    |         |         |         |         |

arl(row) in arl(row).arl(column) has size 22, parameters: 5 5

arl(column) in arl(row).arl(column) has size 11, parameters: 6 6

arl(row).arl(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

\* This job uses all of the 4 processor threads. \*

Note: 1 singularities detected in design matrix. **iterations**

|         |          |     |        |        |
|---------|----------|-----|--------|--------|
| 1 LogL= | -449.818 | S2= | 49.775 | 168 df |
| 2 LogL= | -424.315 | S2= | 40.233 | 168 df |
| 3 LogL= | -405.419 | S2= | 38.922 | 168 df |
| 4 LogL= | -400.288 | S2= | 43.012 | 168 df |
| 5 LogL= | -399.368 | S2= | 47.260 | 168 df |
| 6 LogL= | -399.326 | S2= | 48.388 | 168 df |
| 7 LogL= | -399.324 | S2= | 48.637 | 168 df |
| 8 LogL= | -399.324 | S2= | 48.694 | 168 df |

- - - 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                  |
|----------------------|-------------|----------|----------|----------|----------------------|
| arl(row).arl(column) | 242 effects |          |          |          |                      |
| Residual             | SCA_V 242   | 1.00000  | 48.6942  | 6.81     | 0 P <b>parameter</b> |
| row                  | AR_R 22     | 0.655444 | 0.655444 | 11.62    | 0 P <b>estimates</b> |
| column               | AR_R 11     | 0.437535 | 0.437535 | 5.43     | 0 P                  |

| Source of Variation | Wald F statistics |       |        | P-inc <b>testing</b> |
|---------------------|-------------------|-------|--------|----------------------|
|                     | NumDF             | DenDF | F-inc  |                      |
| 12 mu               | 1                 | 25.0  | 332.04 | <.001 <b>fixed</b>   |
| 1 variety           | 55                | 110.8 | 2.22   | <.001 <b>effects</b> |

Note: The DenDF values are calculated ignoring fixed/boundary/singular variance parameters using algebraic derivatives.

13 mv\_estimates 18 effects fitted

## 14.3 Key output files

```
* This job used at least .2 of the 2.0 Gbyte of primary workspace. *
      6 possible outliers: in section 1 (see .res file)
Finished: 06 Dec 2025 13:56:06.588 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 |
|-----------------------|-------|-----|----------|----------|----------|-----|
| arl(row) .arl(column) |       | 242 | effects  |          |          |     |
| Residual              | SCA_V | 242 | 1.00000  | 48.6942  | 6.81     | 0 P |
| row                   | AR_R  | 22  | 0.655444 | 0.655444 | 11.62    | 0 P |
| column                | AR_R  | 11  | 0.437535 | 0.437535 | 5.43     | 0 P |

| Wald F statistics   |       |       |        |       |
|---------------------|-------|-------|--------|-------|
| Source of Variation | NumDF | DenDF | F_inc  | P_inc |
| 12 mu               | 1     | 25.0  | 332.04 | <.001 |
| 1 variety           | 55    | 110.8 | 2.22   | <.001 |

Note: The DenDF values are calculated ignoring fixed/boundary/singular variance parameters using algebraic derivatives.

13 mv\_estimates 18 effects fitted

Warning: The following screen ignores marginality which should be considered in comparing models involving interactions. It is generally not valid to drop a main effect unless all interactions involving it have also been dropped.

| LINE | REGRESSION        | RESIDUAL          | ADJUSTED            | FACTORS INCLUDED |
|------|-------------------|-------------------|---------------------|------------------|
| NO   | DF SUM OF SQUARES | DF MEAN SQUARE    | R-SQUARED R-SQUARED | 1 12             |
| 1    | 55 0.1747696D+05  | 169 0.7592836D+02 | 0.57663 0.43885     | 55 0             |
|      |                   |                   | *****               | *****            |
| 2    | 56 0.2212823D+05  | 168 0.4869416D+02 | 0.73009 0.64012     | 55 1             |
|      |                   |                   | *****               | *****            |
| 3    | 1 0.1616837D+05   | 223 0.6341022D+02 | 0.53345 0.53136     | 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  $A$  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  $i$ th individual,  $f_i$  is the inbreeding coefficient reported when !DIAG qualifier is given on a pedigree file line,  $1 + f_i$  is

### 14.3 Key output files

the diagonal element of  $A$  and  $\sigma_A^2$  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 in Section 2.4.2.

| Model_Term   | Level   | Effect  | seEffect |                         |
|--------------|---------|---------|----------|-------------------------|
| variety      | LANCER  | 0.000   | 0.000    | variety estimates       |
| variety      | BRULE   | 2.985   | 2.842    |                         |
| variety      | REDLAND | 4.707   | 2.978    |                         |
| variety      | CODY    | -0.3148 | 2.961    |                         |
| :            |         |         |          |                         |
| variety      | NE87615 | 1.034   | 2.934    |                         |
| variety      | NE87619 | 5.938   | 2.850    |                         |
| variety      | NE87627 | -4.377  | 2.998    |                         |
| mu           | 1       | 24.09   | 2.465    | intercept               |
| mv_estimates | 1       | 21.91   | 6.730    | missing value estimates |
| mv_estimates | 2       | 23.22   | 6.722    |                         |
| mv_estimates | 3       | 22.52   | 6.710    |                         |
| mv_estimates | 4       | 23.49   | 6.677    |                         |
| :            |         |         |          |                         |
| mv_estimates | 15      | 29.07   | 4.906    |                         |
| mv_estimates | 16      | 23.96   | 4.577    |                         |
| mv_estimates | 17      | 24.27   | 4.619    |                         |
| 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.

## 14.4 Other ASReml output files

NIN Alliance Trial 1989    Residuals vs Fitted values     $\bar{RvE}_1$   
Residuals (Y) -24.87:15.91    Fitted values (X)    16.77:    35.93

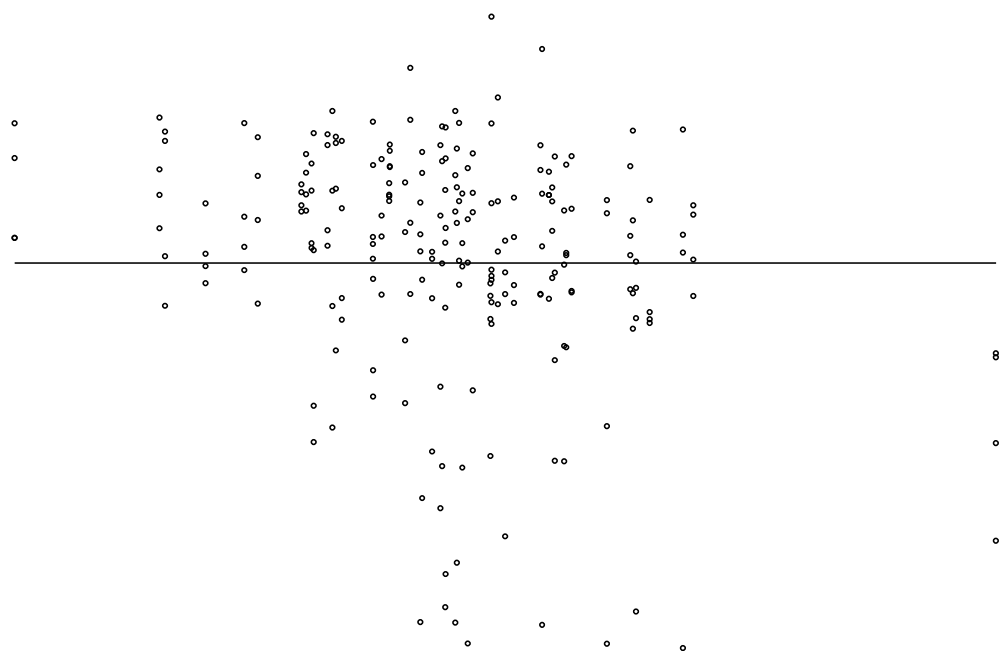


Figure 14.1: Residual versus Fitted values

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      | -0.1000E-36 | -0.1000E-36 | -0.1000E-36 |
| 2      | -0.1000E-36 | -0.1000E-36 | -0.1000E-36 |
| 3      | -0.1000E-36 | -0.1000E-36 | -0.1000E-36 |
| 4      | -0.1000E-36 | -0.1000E-36 | -0.1000E-36 |
| :      |             |             |             |
| 15     | -0.1000E-36 | -0.1000E-36 | -0.1000E-36 |
| 16     | 24.089      | 5.161       | 6.075       |
| 17     | 27.074      | 4.476       | 6.223       |
| 18     | 28.795      | 6.255       | 6.283       |
| 19     | 23.774      | 6.326       | 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      | 0.1478      | 6.159       |
| 242    | 22.464      | 4.436       | 6.605       |

## 14.4 Other ASReml output files

### 14.4.1 The .aoV file

## 14.4 Other ASReml output files

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  $\lambda$
- $\lambda$  as defined in Kenward and Roger (1997).

```
Incremental Wald F statistics - calculation of Denominator degrees of freedom
Source          Size NumDF   F-value  Lambda*F   Lambda   DenDF
mu              1      1  332.0393  332.0393   1.0000   25.0233
variety        56     55   2.2253   2.2242   0.9995  110.8296
```

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 `!DEFINE` qualifier defines the model term `linNitr` as the linear covariate representing nitrogen 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
!DEFINE linNitr nitrogen 0.6 0.4 0.2 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
Coefficient of Determination: NDF 11.00 DenDF 42.79 Fall 10.71 CD 73.36
```

```
Approximate stratum variance decomposition
Stratum      Degrees-Freedom  Variance      Component Coefficients
idv(blocks)      5.00      3175.08      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
Coefficient of Determination: NDF 11.00 DenDF 42.79 Fall 10.71 CD 73.36
```

```
Model_Term          IDV_V      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
units                SCA_V      72     1.00000     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
Note: The DenDF values are calculated ignoring fixed/boundary/singular

```

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).

```

This file reports details concerning the calculation and testing of the
Wald F-statistics reported in the .asr file.

```

```

Table showing the reduction in the numerator degrees of freedom
for each term as higher terms are absorbed.

```

```

Source          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

```

```

-- Sources --
Source      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  *  .  .

```



## 14.4 Other ASReml output files

---

```
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 Wald F statistics - calculation of Denominator degrees of freedom
Source          Size NumDF   F-value   Lambda*F   Lambda   DenDF
mu                1     1  245.1410   245.1410   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 Wald F statistics - calculation of Denominator degrees of freedom
Source          Size NumDF   F-value   Lambda*F   Lambda   DenDF
mu                1     1  138.1361   138.1361   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 low-level debugging 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 .srs file

The .srs file contains the data and residuals from the analysis in single precision binary form. The file is produced when the !RES qualifier (Table 5.8) is invoked 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.
#
Response yield
# Fields are:
# GN, Term, Type, PSpace, Initial_value, RP_GN, RP_scale.
#
    4, "Variance 1", V, P,    1.0000000    ,    4,    1
    5, "ar1(row).ar1(column);ar1(row)_1", R, P,    0.65544434    ,    5,    1
    6, "ar1(row).ar1(column);ar1(column)_1", R, P,    0.43753461    ,    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.
#
# This file is overwritten if not being read.
```

The Response line allows ASReml to check, when the ASReml .msv file is read, that the values pertain to the same trait. If not, the file will not be used. Delete the Response line to allow the initial values to be used though the analysis is for a different trait.

### 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 W directive in VPREDICT (see VPREDICT in [Section 13.3](#)) is used to specify filename and functions. This is nincov.pin generated by using in file nin89a.as.

```
VPREDICT !DEFINE
X COV12 arlv(row).ar1(column);Residual* arlv(row).ar1(column);arlv(row) # row covariance
X COV13 arlv(row).ar1(column);Residual* arlv(row).ar1(column);ar1(column) # column cov.
X COV23 COV12*arlv(row).ar1(column);ar1(column)
W nincov COV # Prints to nincov.vpc and nincov.vpv
```

And the output presented from the file .pvc file is

```
ar1(row).ar1(column)          242 effects
  1 ar1(row).ar1(column);Residual      V  242      48.6942      7.15040
  2 ar1(row).ar1(column);ar1(row)      R   22      0.655444      0.564065E-01
  3 ar1(row).ar1(column);ar1(column)    R   11      0.437535      0.805774E-01
X COV12 ar1(row).ar1(column);Residual*ar1(row).ar1(column);ar1(row)
  4 COV12      = ar1(row).ar1      1 x ar1(row).ar1      2 =      31.9163      6.5876
X COV13 ar1(row).ar1(column);Residual*ar1(row).ar1(column);ar1(column)
  5 COV13      = ar1(row).ar1      1 x ar1(row).ar1      3 =      21.3054      5.2982
```

## 14.4 Other ASReml output files

```
X COV23 COV12*ar1(row).ar1(column);ar1(column)
  6 COV23      = COV12    4 x ar1(row).ar1    3 =      13.9645    3.6639
W nincov COV
VVP matrix for parameters 4 to 6 written to nincov.vpv
      with values and labels in nincov.vpc
Note: The parameter estimates are followed by
```

### 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                                06 Dec 2025 14:04:22 title line
                                                    nin89AR

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              24.0886          2.4647 E
BRULE             27.0736          2.4946 E
REDLAND           28.7952          2.5066 E
CODY              23.7738          2.4972 E
...
NE87613          26.2111          2.4635 E
NE87615          25.1224          2.4436 E
NE87619          30.0263          2.4668 E
NE87627          19.7113          2.4835 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
- Legendre polynomials produced by `leg()` model term
- Orthogonal polynomials produced by `pol()` model term
- The design matrix formed for the `spl()` model term

## 14.4 Other ASReml output files

---

- Predicted values of the curvature component of cubic smoothing splines
- The empirical variance-covariance matrix based on the **BLUPs** when a  $\Sigma \otimes I$  or  $I \otimes \Sigma$  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
- An eigen analysis of each variance matrix 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 matrix 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  $\log(\text{absolute residual})$  on  $\log(\text{predicted value})$  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

228

## 14.4 Other ASReml output files

```
SLOPES FOR LOG(ABS(RES)) on LOG(PV) for Section 1
0.15
SLOPES FOR LOG(SDi) on LOG(PVBari) for Section 1
1.37
```

Histogram of RvE\_1\_A: MaxFreq 17, Range -24.873 15.915

```

          *
          *
        *  ** ***
        *  ** *** *
        ** *** ** *
        *****
        *****
        *****
*  *
*  ** ** ** ** ** ***** ** *
```

Min Mean Max -24.873 0.27965 15.915 omitting 18 zeros

Spatial diagnostic statistics of ar1(row).ar1(column) 22 11

Residual Plot and Autocorrelations

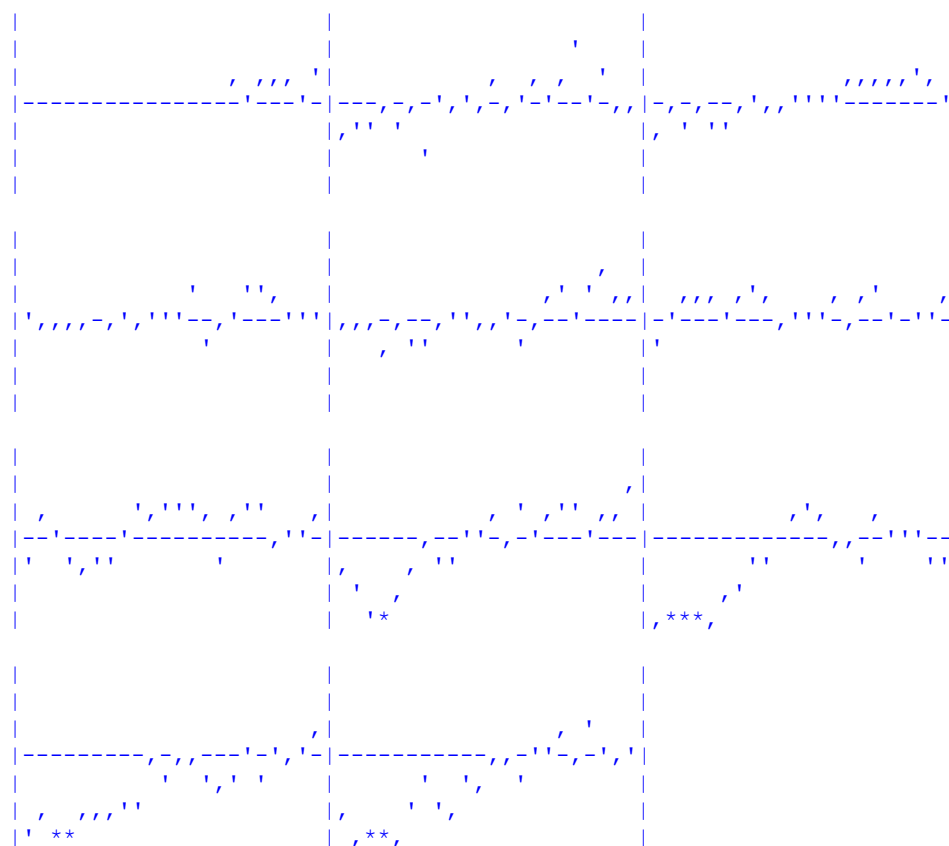
<LOo- +xXH> [se 0.077]

```
|      +xxxxX|
|--- - 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+xH|
|<<<<<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 = 6.978 ]

|      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |
|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|
| 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 | -184 | -179 | -189 | -118 | -124 | 14   | -53  | 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 (of 11), row 4 (of 22)] is -3.32 SD  
 Residual [section 1, column 9 (of 11), row 2 (of 22)] is -3.33 SD  
 Residual [section 1, column 9 (of 11), row 3 (of 22)] is -3.52 SD  
 Residual [section 1, column 10 (of 11), row 3 (of 22)] is -3.56 SD  
 Residual [section 1, column 10 (of 11), row 4 (of 22)] is -3.35 SD  
 Residual [section 1, column 11 (of 11), row 3 (of 22)] is -3.52 SD  
 6 possible outliers in section 1: test value 23.027795523

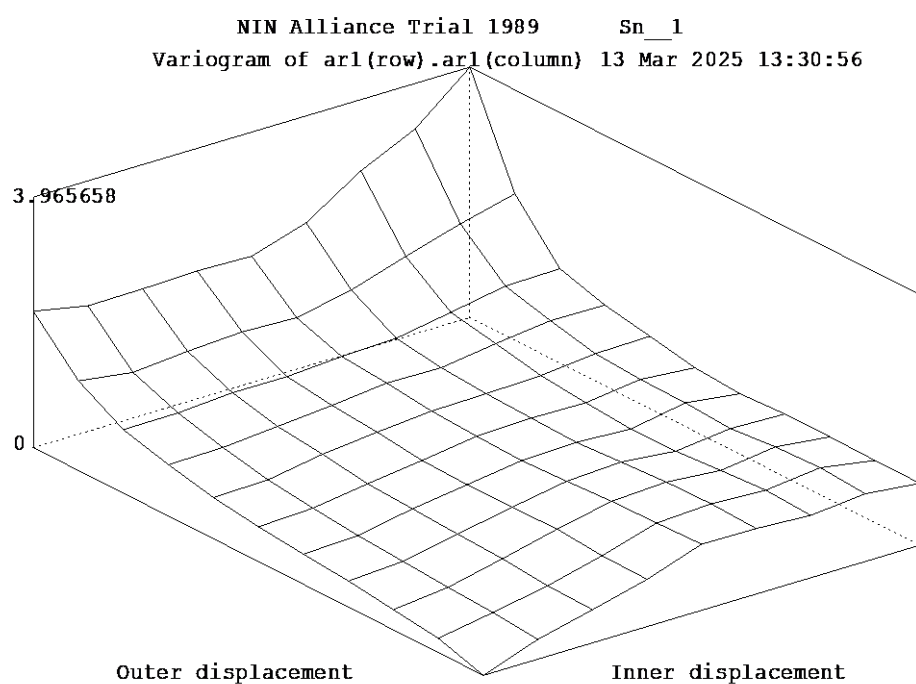


Figure 14.2: Variogram of residuals

Figure 14.2 to Figure 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 5.7). 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.

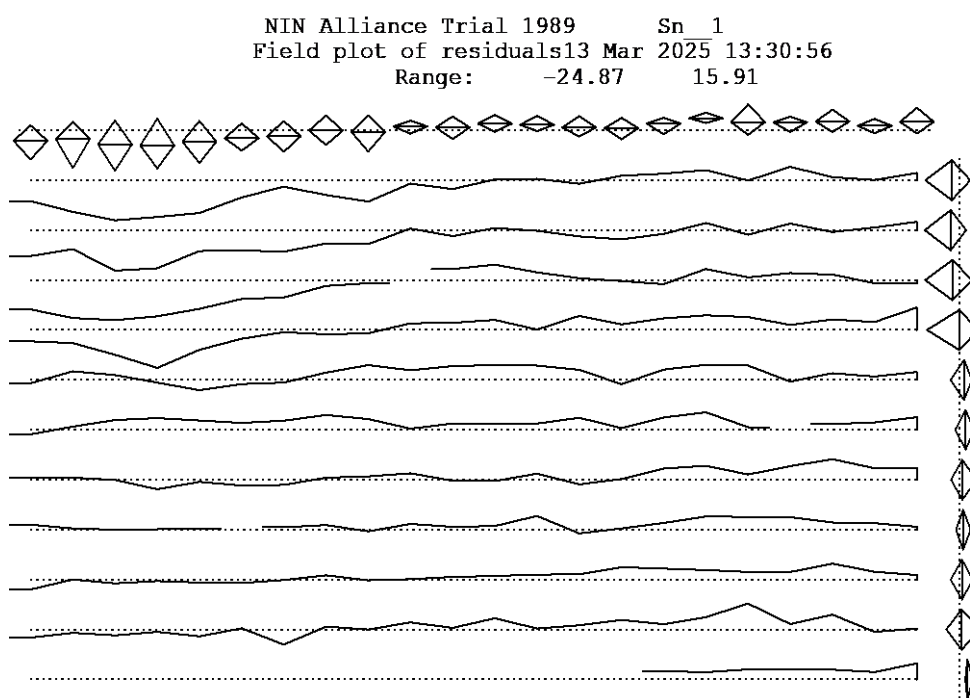


Figure 14.3: Plot of residuals in field plan order



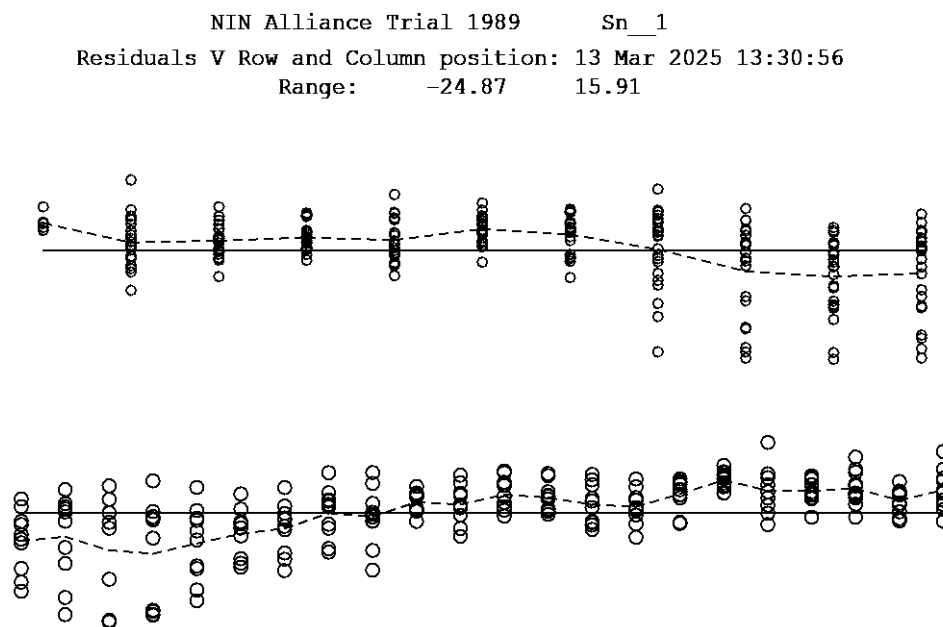


Figure 14.4: Plot of the marginal means of the residuals

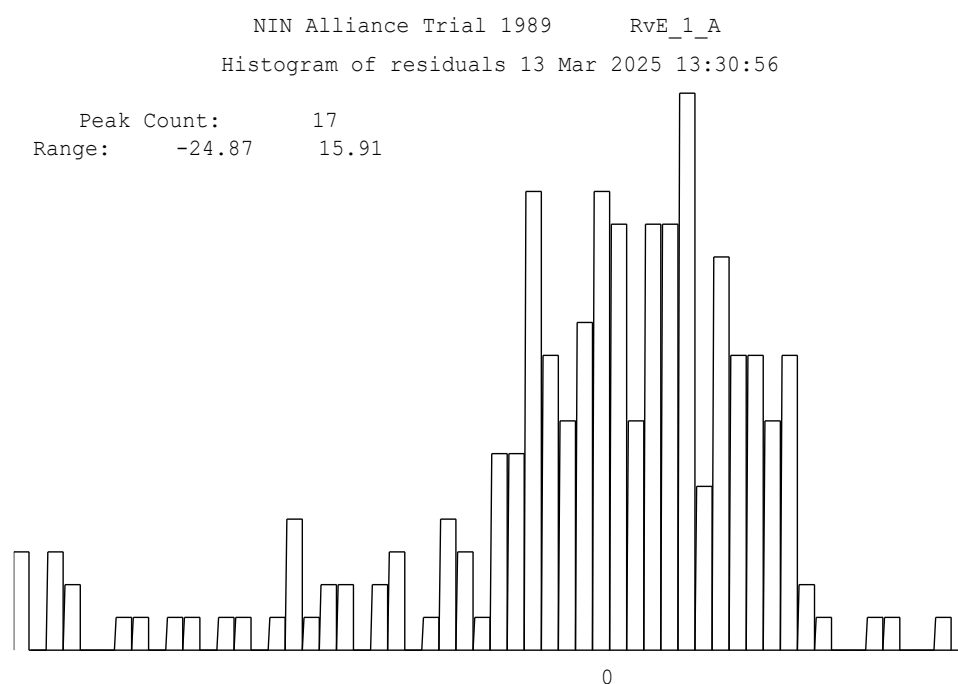


Figure 14.5: Histogram of residuals

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

DISPLAY of variance partitioning for XFA structure in xfa2(Env).Geno



|         |       |
|---------|-------|
| NE83404 | 27.39 |
| NE83406 | 24.28 |
| NE83407 | 22.69 |
| CENTURA | 21.65 |
| SCOUT66 | 27.52 |
| COLT    | 27.00 |
| :       |       |
| NE87613 | 29.40 |
| NE87615 | 25.69 |
| NE87619 | 31.26 |
| NE87627 | 23.23 |

### 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.
#
Response yield
# Fields are:
# GN, Term, Type, PSpace, Initial_value, RP_GN, RP_scale.
#
    4, "Variance 1", V, P,    1.0000000    ,    4,    1
    5, "ar1(row).ar1(column);ar1(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.
#
# This file is overwritten if not being read.
```

### 14.4.11 The .vpc and .vpv files

The .vpc and .vpv files contains variance component parameters in the .vpc file and estimates of variances and covariances of the component parameters in the .vpv file. These files are written when the VPREDICT command W (see VPREDICT in Section 13.3) is used.

For example, the command

```
VPREDICT !DEFINE
W varcov 1:3
```

Creates the file `varcov.vpc` containing

```
48.6942      ar1(row).ar1(column);Residual      V   242
0.655444      ar1(row).ar1(column);ar1(row)      R    22
0.437535      ar1(row).ar1(column);ar1(column)    R    11
```

and file `varcov.vpv` containing

```
51.1553
0.217463      0.317920E-02
0.675012E-01 -0.201140E-02  0.649554E-02
```

### 14.4.12 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 5.8) 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

$$\begin{bmatrix} \sigma^2 & \cdot \\ \tilde{\beta}_D & \sigma^2 \mathbf{C}^{DD} \end{bmatrix}$$

where  $\tilde{\beta}_D$  is the dense portion of  $\beta$  and  $\mathbf{C}^{DD}$  is the dense portion of  $\mathbf{C}^{-1}$ . This is part of `nin89a.vrb`. Note that the first element is the estimated error variance, that is, 48.6942, see the variance component estimates in the `.asr` output.

```
0.486942E+02  0.000000E+00  0.000000E+00  0.298503E+01  0.000000E+00
0.807428E+01  0.470659E+01  0.000000E+00  0.456582E+01  0.886569E+01
-0.314807E+00  0.000000E+00  0.409981E+01  0.476505E+01  0.876618E+01
0.295389E+01  0.000000E+00  0.343281E+01  0.389572E+01  0.416094E+01
0.743507E+01  0.163168E+01  0.000000E+00  0.377119E+01  0.428051E+01
0.472477E+01  0.402657E+01  0.837160E+01  0.129022E+01  0.000000E+00
0.330012E+01  0.347413E+01  0.357562E+01  0.316872E+01  0.412076E+01
0.768166E+01  0.309427E+00  0.000000E+00  0.376584E+01  0.419734E+01
0.395660E+01  0.383390E+01  0.458412E+01  0.378522E+01  0.985052E+01
0.226429E+01  0.000000E+00  0.379226E+01  0.442405E+01  0.439439E+01
0.402458E+01  0.440488E+01  0.362343E+01  0.502043E+01  0.901083E+01
0.508523E+01  0.000000E+00  0.393560E+01  0.430453E+01  0.423711E+01
0.428778E+01  0.417814E+01  0.363292E+01  0.444739E+01  0.527225E+01
0.855119E+01  0.243603E+01  0.000000E+00  0.351319E+01  0.369932E+01
:
```

For clarity, the first 5 rows of the lower triangular matrix are

```
48.6942
0.0000      0.0000
2.9850      0.0000  8.0743
4.7066      0.0000  4.5658  8.8657
-0.3148      0.0000  4.0998  4.7651  8.7662
```

### 14.4.13 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
51.1553
0.217463      0.317920E-02
0.675012E-01 -0.201140E-02  0.649554E-02
```

### 14.4.14 The .wvr file

The .wvr file contains working variables. The matrix is printed with one element per row and indexed by the row and variance parameters. This file uses the same order for the variance parameters as printed in the .asr file. This is part of nin89a.wvr with the parameter estimates in the order: error variance, spatial row correlation, spatial column correlation.

```
ROW  VPAR  Working_Variable
1      1 -0.100000E-36
1      2 -0.243920
1      3 -6.05493
2      1  19.4000
2      2 -0.906653
2      3 -2.39839
3      1  29.8500
3      2  4.64221
3      3  2.62897
:
241    1  27.4500
241    2 -4.26923
241    3  5.96033
242    1  26.9000
242    2  1.16402
242    3  5.28910
```

## 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 <i>fixed</i> part of the model. These provide for an incremental or optionally a conditional test of significance (see Section 6.11).   |
| data summary  | .asr file<br>.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/variante field distinction.<br><br>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 (eigenvalues and eigenvectors) to the .res file.   |
| elapsed time  | .asr file<br>.asl file | this can be determined by comparing the start time with the finishing time.<br><br>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 $\log(\text{abs}(\text{residual}))$ against $\log(\text{predicted value})$ . 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. |
| 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          |  |

## 14.5 ASReml output objects and where to find them

---

| output object                                | found in               | comment  |
|--|------------------------|--|
| 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<br>.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<br>.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          |  |

---

# 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 Log-likelihood 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 contact support at <https://vsni.co.uk/support>.

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

- Variable name has been misspelt; variable names are case sensitive



## 15.2 Common problems

---

- 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.

The primary file to examine for diagnostic messages is the `.asr` file. When **ASReml** finds something atypical or inconsistent, it prints a 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 a 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.

## 15.3 Things to check in the .asr file

If the problem is not resolved after these checks, you may need to contact Customer Support at <https://vsni.co.uk/support/>. 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 variety
```

```
ASReml 4.3ni [07 Mar 2025] NIN Alliance Trial 1989
Windows x64      2.0 Gbyte  nin89_ERR1  14 Mar 2025 22:26:32.055
* Licensed to: valid
* Your ASReml license expires in 31 days *
*****
* Contact https://vsni.co.uk/support for licensing and support *
***** ARG *
Folder: C:\NIN\NIN89
  There is no file called nin9.asd
  in the current working folder:
  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.

Error: No data file found in working folder.
Error: No data file line detected.

***** ***** Dump of Model structures ***** *****
Fault: [11] Error parsing yield ~ mu variety !R Repl
Last line read was:  yield ~ mu variety !R Repl
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
nin89_ERR1 C:\Users\sgeza\OneDrive\Desktop\ASReml
  11 factors defined [max7500].
    0 variance parameters [max2500].    2 special structures
Last line read was:  yield ~ mu variety !R Repl
Finished:      14 Mar 2025 22:26:33.365  Error parsing yield ~ mu variety !R Repl
```

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

- Filename misspelt; there is no file `nin9.asd` in the working folder
- Unrecognised qualifier (should be `!SKIP`)
- 'Variety' has alphabetic level labels but not declared has such; `!A` required
- COMMA missing from first line of model; `!R Repl` is part of the model but not recognised as such
- Misspelt variable label in linear model; `Repl` should be `repl`
- Misspelt variable labels in residual model
- The data has missing cells with respect to the declared residual structure
- 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 11 !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.

```
...
QUALIFIERS: !SLIP 1
Warning: Unrecognised qualifier at character 11 !SLIP 1
         It may need to be on the next line!
Reading nin89.asd  FREE FORMAT skipping      0 lines

Univariate analysis of yield
Note: 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

*****          *****  Dump of Model structures  *****          *****
Fault: [0] Missing/faulty !SKIP or !A needed for variety
Last line read was:  variety id pid raw rep nloc yield lat long row column
nin89_ERR1 nin89.asd
Model specification:  TERM LEVELS GAMMAS
mu                                0
variety                          0
12 factors defined [max7500].
0 variance parameters [max2500]. 2 special structures
Last line read was:  variety id pid raw rep nloc yield lat long row column
Finished:   14 Mar 2025 22:33:25.119  Missing/faulty !SKIP or !A needed for variety
```

```
NIN Alliance Trial 1989
variety *
...
row * column *
nin89.asd !SLIP 1
yield ~ mu variety
!r Repl
residual ar1(Row).ar1(Col)
PREDICT varierty
```

## 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.

```
...
QUALIFIERS: !SKIP 1
Reading nin89.asd  FREE FORMAT skipping      1 lines

Univariate analysis of yield
Note: 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

*****          *****  Dump of Model structures  *****          *****
Fault: [0] 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
nin89_ERR1 variety id      pid      raw      rep      nloc      yield lat
```

```
NIN Alliance Trial 1989
variety *
:
row * column *
nin89.asd !SKIP 1
yield ~ mu variety
!r Repl
residual ar1(Row).ar1(Col)
PREDICT varierty
```

## 15.4 An example

```

Model specification:  TERM LEVELS GAMMAS
mu                  0
variety             0
  12 factors defined [max7500].
  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:    14 Mar 2025 22:37:21.394  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.

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.

```

NIN Alliance Trial 1989
variety !A
:
row * column *
nin89.asd !SKIP 1
yield ~ mu variety
!r Repl
residual ar1(Row).ar1(Col)
predict varierty

```

```

:
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      0  1.000      28.50      56.00      16.20
3 pid               0      0      0  1101.      2628.      4156.      1121.
4 raw               0      0      0  21.00      510.5      840.0      149.0
5 repl              4      0      0      1      2.5000      4
6 nloc              0      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      0  4.300      27.22      47.30      12.90
9 long              0      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

***** ***** *****  Dump of Model structures  ***** ***** *****
Fault: [0] Error in variance header line: !R Repl
Last line read was:  !R Repl 0 0 0 0
nin89_ERR1 variety id      pid      raw      rep      nloc      yield  lat
Model specification:  TERM LEVELS GAMMAS
variety              56
mu                   1
  12 factors defined [max7500].
  0 variance parameters [max2500].  2 special structures
Final parameter values [ 2:  0]
Last line read was:  !R Repl 0 0 0 0
Finished:    17 Mar 2025 12:33:03.804  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.

```

:
QUALIFIERS: !SKIP 1
Reading nin89.asd  FREE FORMAT skipping      1
lines
Model term "Repl" is not valid/recognised.

Warning: Variable Repl not yet defined

*****          *****  Dump of Model structures  *****          *****
Fault: [0]  Error processing model terms
  Last line read was:      Repl
nin89_ERR1 variety      id      pid      raw      rep      nloc      yield lat
  Model specification:  TERM LEVELS GAMMAS
mu                                0
variety                          0
  12 factors defined [max7500].
  0 variance parameters [max2500].  2 special structures
  Last line read was:      Repl

Warning: Variable Repl not yet defined
Finished:   17 Mar 2025 12:38:56.542      Error processing model terms

```

```

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

```

## 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.

```

:
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      0    1.000    28.50      56.00    16.20
...
10 row           22      0      0        1    11.7321      22
11 column         11      0      0        1     6.3304      11
12 mu              1

```

```

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

```

## 15.4 An example

```
Model term "ar1(Row).ar1(Col)" is not valid/recognised.
```

```
Error: Row not found for ar1(Row).ar1(Col)
```

```
Error: ar1(Row).ar1(Col) is not successfully parsed.
```

```
***** Dump of Model structures *****
Fault: [1] Error: Failed to parse RESIDUAL line.
Last line read was: Residual ar1(Row).ar1(Col)
nin89_ERR1 variety      id      pid      raw      rep      nloc      yield lat
Model specification:  TERM LEVELS GAMMAS
variety                                56
mu                                    1
repl                                4      0.100 [ 3]
12 factors defined [max7500].
0 variance parameters [max2500]. 2 special structures
Final parameter values [ 3: 0]
Last line read was: Residual ar1(Row).ar1(Col)

Error: Row not found for ar1(Row).ar1(Col)
Error: ar1(Row).ar1(Col) is not successfully parsed.
Finished: 17 Mar 2025 12:43:43.481 Error: Failed to parse RESIDUAL line.
```

## 7. Missing plots in field layout

The variables `row` and `column` define a  $22 \times 11$  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)
```

```
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 variety
```

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
```

```
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    |
| :                             |              |             |       |         |         |         |          |
| 11 column                     | 11           | 0           | 0     | 1       | 6.3304  | 11      |          |
| 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       |         |         |          |

## 15.5 Information, Warning and Error messages

---

```
ar1(row).ar1(col) [ 4: 6] initialized.
Total Records defined: 242; Records in data: 224
242
Forming 61 equations: 57 dense.
Initial updates will be shrunk by factor 0.316

Note: Invalid argument, unrecognised qualifier or
      vector space exhausted at 'varierty '
Failed to parse/process line: PREDict varierty

Error: R structures do not match records in data.
Error: Spatial Layout is not rectangular grid
Error: varierty is not successfully parsed.

***** ***** ***** Dump of Model structures ***** ***** *****
Fault: [44] Variance structure does not match data
Last line read was: !
nin89_ERR1 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 [max7500].
6 variance parameters [max2500]. 2 special structures
Final parameter values [ 3: 6] 0.10000 1.0000 0.10000
0.10000
Last line read was: !
Finished: 17 Mar 2025 12:47:47.352 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

```
Note: Invalid argument, unrecognised qualifier or
      vector space exhausted at 'varierty '
Failed to parse/process line: 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(are) 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



## 15.5 Information, Warning and Error messages

- 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 and to report problems to the link <https://vsni.co.uk/support>.

Table 15.1: Some information messages and comments

| information message                                    | comment   |
|--|---|
| <code>Logl converged</code>                            | The REML log-likelihood last changed less than 0.002 * iteration number and variance parameter values appear stable.  |
| <code>BLUP run done</code>                             | A full iteration has not been completed. See discussion of <code>!BLUP</code> .   |
| <code>JOB ABORTED by USER</code>                       | See discussion of <code>ABORTASR.NOW</code> .   |
| <code>Logl converged, parameters not converged</code>  | The change in REML log-likelihood was small and convergence was assumed but the parameters are, in fact, still changing.  |
| <code>Logl not converged</code>                        | 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 <code>!CONTINUE</code> command line option (see Section 11.3 on job control), or to change the model and/or initial values before proceeding. The sequence of estimates is reported in the <code>.res</code> file. It may be necessary to simplify the model and estimate the dominant components before estimating other terms if the LogL is oscillating. |
| <code>Warning: Only one iteration performed</code>     | Parameter values are not at the REML solution.  |
| <code>Parameters unchanged after one iteration.</code> | Parameters appear to be at the REML solution in that the parameter values are stable.   |

Messages beginning with the word **Note:** 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

## 15.5 Information, Warning and Error messages

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.

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: <i>e</i> missing values generated by <code>!^</code> transformation | All data values should be positive.   |
| Warning: <i>i</i> singularities in AI matrix                                 | Usually means the variance model is overparameterized. Look up <code>!AISING</code> .   |
| Warning: <i>m</i> variance structures were modified                          | The structures are probably at the boundary of the parameter space.   |
| Warning: <i>n</i> missing values were detected in the design                 | Either use <code>!MVINCLUDE</code> or delete the records.   |
| Warning: <i>n</i> negative weights   | It is better to avoid negative weights unless you can check ASReml is doing the correct thing with them.  |
| Warning: <i>r</i> 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 <i>term</i> has more levels [ ## ] than expected [ ## ]:             | You have probably mis-specified the number of levels in the factor or omitted the <code>!I</code> qualifier (see Section 5.4.1 on data field definition syntax). ASReml corrects the number of levels.  |
| Warning: <i>term</i> in the PREDICT <code>!IGNORE</code> list                | The term did not appear in the model.   |
| Warning: <i>term</i> in the PREDICT <code>!USE</code> list                   | The term did not appear in the model.   |
| Warning: <i>term</i> is ignored for prediction                               | Terms like <code>units</code> and <code>mv</code> cannot be included in prediction.   |
| Warning: Check if you need the <code>!RECODE</code> qualifier                | <code>!RECODE</code> 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 ( <code>!GP</code> )                   | Suggest drop the term and refit the model.  |
| Warning: Dropped records were not evenly distributed across                  | <code>!MVREMOVE</code> 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

---

|  |   |
|--|---|
| 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: Fixed levels for factor                                       | User nominated more levels than are permitted.  |
| Warning: Initial gamma value is zero                                   | Constraint parameter is probably wrongly assigned.  |
| 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: 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                            | This occurs because it contains errors.   |
| Warning: Spatial mapping information for side                          | This gives details so you can check ASReml is doing what you intend.  |
| Warning: Standard errors   | That is, these standard errors are approximate.   |
| 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 !X !Y !G qualifiers are ignored. There is no data to plot | Revise the qualifier arguments.   |
| 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.   |

## 15.5 Information, Warning and Error messages

|   |   |
|---|---|
| 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<br>!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:<br>ASReml failed in ... | The job file should be in ASCII format.<br>Try running the job with increased workspace, or using a simpler model. Otherwise send the job to VSNi 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<br>Check IDV structure     | An argument in the CALC statement is not valid.<br>ASReml is using IDV variance structure but wonders whether that is what you intended.  |
| Context of read error Data<br>Error: At record ...    | ASReml found alpha characters 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.  |

## 15.5 Information, Warning and Error messages

| error message  | probable cause/remedy  |
|--|--|
| Convergence failed   | <p>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</p> <ul style="list-style-type: none"> <li>• Define US structures as positive definite by using !GP.</li> <li>• Supply better starting values, fix parameters that you are confident of while getting better estimates for others (that is, fix variances when estimating covariances).</li> <li>• Fit a simpler model.</li> <li>• Reorganise the model to reduce covariance terms (for example, use CORUH instead of US) .</li> </ul> |
| 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 !DEFINE 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 failed to find an order for solving the mixed model equations. See !EQORDER for some discussion. Check the model. Try a simpler model. Try increasing !WORKSPACE  |
| Error parsing  | This error comes from the main read routine! or from the variable definition parsing routine.  |

## 15.5 Information, Warning and Error messages

| error message   | probable cause/remedy  |
|---|--|
| Error reading <i>something</i>  | There are several messages of this form where <i>something</i> is what ASReml is attempting to read. Either there is an error telling ASReml to read <i>something</i> when it does not need to, or there is an error in the way <i>something</i> is specified. |
| Error reading the data:   | The data file could not be interpreted: alphanumeric fields need the !A qualifier.   |
| Error reading the DATA<br>FILENAME line                                       | Data file name may be wrong.   |
| Error reading the model factor<br>list  | The model specification line is in error: a variable is probably misnamed.   |
| Error: Ran out of space to<br>code records to sort them!'/                    | Declare the levels in the !ROWFACTOR, !COLUMNFACTOR and !SECTION variables more accurately.  |
| Error setting constraints<br>(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<br>variable   | The specified dependent variable name is not recognised.   |
| Error setting MBF design<br>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<br>size:   | The declared size of the error structures does not match the actual number of data records.  |
| Error when reading knot point<br>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<br>line<br>Failed to read R/G structure<br>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<br>files   | Check your MYOWNGDG program and the .gdg file.   |
| Failed when sorting pedigree<br>...   | Maybe increase !WORKSPACE. Messages may identify a problem with the pedigree.  |
| Failed when processing<br>pedigree file ...                                   | Preceding messages should indicated the problems encountered.  |
| Failed while ordering<br>equations.   | This indicates the job needs more memory than was allocated or is available. Try increasing the workspace or simplifying the model.  |

## 15.5 Information, Warning and Error messages

| error message                                     | probable cause/remedy   |
|---|---|
| FORMAT error reading ...                          | Likely causes are <ul style="list-style-type: none"><li>• Bad syntax or invalid characters in the variable names; variable names must not include any of these symbols ; ! -(:#\$ and ..</li><li>• The data file name is misspelt.</li><li>• There are too many variables declared or there is no valid <i>value</i> supplied with an arithmetic transformation option.</li></ul> |
| 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.   |
| 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 <i>generated</i> 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 <i>model specification</i> 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.   |



## 15.5 Information, Warning and Error messages

| error message                                  | probable cause/remedy   |
|--|---|
| 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.   |
| Matérn: ...                                    | Numerical problems calculating the Matérn function. If rescaling the <i>X</i> , <i>Y</i> coordinates 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.   |
| 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 increasing WORKSPACE, or predicting in parts.  |



## 15.5 Information, Warning and Error messages

| error message                                 | probable cause/remedy   |
|---|---|
| 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 <i>g</i> on the variance header line for the number of <i>G structures</i> (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 VSNi.   |
| 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 VSNi 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 are deleted or the model fully fits the data.   |
| R header SECTIONS DIMNS<br>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 header SITE DIM<br>GSTRUCT        |   |
| Variance header: SEC DIM<br>GSTRUCT           |   |

## 15.5 Information, Warning and Error messages

| error message                                     | probable cause/remedy   |
|---|---|
| R structure error ORDER<br>SORTCOL MODEL GAMMAS:  | An error reading the error model.   |
| R structures are larger than<br>number of records | Maybe you need to include <code>mv</code> in the model to stop <b>ASReml</b> discarding records with missing values in the response variable.   |
| REQUIRE !ASUV qualifier for<br>this R structure   | Without the <code>ASUV</code> qualifier, the multivariate error variance <b>MUST</b> be specified as <code>US</code> .  |
| REQUIRE I x E R structure<br>Scratch:             | Apparently <b>ASReml</b> could not open a scratch file to hold the transformed data. On UNIX, check the temp directory <code>//tmp</code> for old large scratch files.  |
| 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 <code>!WORKSPACE</code> (see Section 11.3 on memory) command line option. Otherwise, you may need to send your data and the <code>.as</code> files to Customer Support for debugging.  |
| Singularity appeared in AI<br>matrix              | See the discussion on <code>!AISINGULARITIES</code> .   |
| Singularity in Average<br>Information Matrix      |   |
| SINGULARITY IN ...                                | Problem performing the 'Regression Screen'.   |
| Sorting data by !Section !Row<br>...              | 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.   |
| Sorting the data into field<br>order              |   |
| STOP SCRATCH FILE DATA STORAGE<br>ERROR:          | <b>ASReml</b> 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 <code>!NOSCRATCH</code> 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<br>level labels:     | If the factor level labels are actually all integers, use the <code>!I</code> option instead. Otherwise, you will have to convert a factor with alphanumeric labels to numeric sequential codes external to <b>ASReml</b> so that an <code>!A</code> option can be avoided.   |
| Too many factors with !A or<br>!I; max 100        | The data file may need to be rewritten with some factors recoded as sequential integers.  |
| Too many [max 20] dependent<br>variables          | This is an internal limit. Reduce the number of response variables. Response variables may be grouped using the <code>!G</code> factor definition qualifier so that more than 20 actual variables can be analysed.  |
| Unable to invert R or G [US?]<br>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 ( <code>US</code> , <code>CHOL</code> and <code>ANTE</code> models). It may also occur if an <i>identity by unstructured</i> ( <code>ID⊗US</code> ) error variance model is not specified in a multivariate analysis (including <code>!ASMV</code> ), 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 <code>!GP</code> to force the matrix positive definite, and try reducing the updates by using the <code>!STEP</code> qualifier. Otherwise, you could try fitting an alternative parameterisation. |

## 15.5 Information, Warning and Error messages

---

| error message                                  | probable cause/remedy   |
|--|---|
| Unable to invert R or G<br>[CORR?] matrix:     | Generally refers to a problem setting up the mixed model equations.<br>Most commonly, it is caused by a non-positive definite matrix. |
| Variance structure is not<br>positive definite | Use better initial values or a structured variance matrix that is<br>positive definite.   |
| XFA model not permitted in R<br>structures     | You may use FA or FACV. The R structure must be positive<br>definite.   |
| XFA may not be used as an R<br>structure       |   |

# 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

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

## 16.2 Split plot design – Oats

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 subplots, 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 analysis - oat
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 variety # Print table of predicted means
PREDICTnitrogen variety !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
Coefficient of Determination: NDF 11.00 DenDF 42.79 Fall 10.71 CD 73.36

Approximate stratum variance decomposition
Stratum      Degrees-Freedom   Variance      Component Coefficients
idv(blocks)      5.00      3175.08      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
Coefficient of Determination: NDF 11.00 DenDF 42.79 Fall 10.71 CD 73.36

Model_Term          IDV_V      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
units                 SCA_V      72     1.00000     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          5.0      245.14     245.14 . <.001
4 variety                  2          10.0      1.49      1.49 A 0.272
2 nitrogen                  3          45.0      37.69     37.69 A <.001
9 variety.nitrogen         6          45.0      0.30      0.30 B 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 component for each term in the random model in the columns labelled `Gamma` and `Component`, respectively. The stratum variance table shows the link between the variance components (random terms involving blocks) and the mean squares and degrees of freedom that would be obtained if these terms were fitted as fixed effects.

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

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

The 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  $\chi^2$  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.

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:  nitrogen
```

## 16.3 Unbalanced nested design – Rats

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

----- 2 -----  
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 !SKIP 1 !FCON !DOPATH $1
!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.



## 16.3 Unbalanced nested design – Rats

```

1 LogL= 74.2174      S2= 0.19670      315 df    0.1000
2 LogL= 79.1552      S2= 0.18752      315 df    0.1487
3 LogL= 83.9384      S2= 0.17755      315 df    0.2446
4 LogL= 86.4471      S2= 0.17054      315 df    0.3801
5 LogL= 87.1964      S2= 0.16640      315 df    0.5292
6 LogL= 87.2396      S2= 0.16537      315 df    0.5828
7 LogL= 87.2398      S2= 0.16530      315 df    0.5867
8 LogL= 87.2398      S2= 0.16530      315 df    0.5867
Final parameter values                                0.5867

- - - Results from analysis of weight - - -
Akaike Information Criterion      -170.48 (assuming 2 parameters).
Bayesian Information Criterion    -162.97
Coefficient of Determination: NDF 6.00 DenDF 71.30 Fall 18.51 CD 60.90

Approximate stratum variance decomposition
Stratum      Degrees-Freedom    Variance      Component Coefficients
idv(dam)          22.56      1.27763      11.5      1.0
Residual Variance 292.44      0.165300      0.0      1.0
Coefficient of Determination: NDF 6.00 DenDF 71.30 Fall 18.51 CD 60.90

Model_Term              Gamma      Sigma      Sigma/SE      % C
idv(dam)                IDV_V    27      0.586674      0.969770E-01      2.92      0 P
units                   SCA_V    322      1.00000      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 . <.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 0.672
Note: The DenDF values are calculated ignoring fixed/boundary/singular
      variance parameters using algebraic derivatives.
9 idv(dam)                27 effects fitted
* This job used at least .2 of the 2.0 Gbyte of primary workspace. *
SLOPES FOR LOG(ABS(RES)) on LOG(PV) for Section 1
2.27
3 possible outliers in Section 1: 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

## 16.3 Unbalanced nested design – Rats

'dropping' unit 66 by fitting an effect for it with the fixed effect term `out(66)`.

We refit the model without the `dose.sex` term (part 3). Note that the variance parameters are re-estimated, though there is little change from the previous analysis.

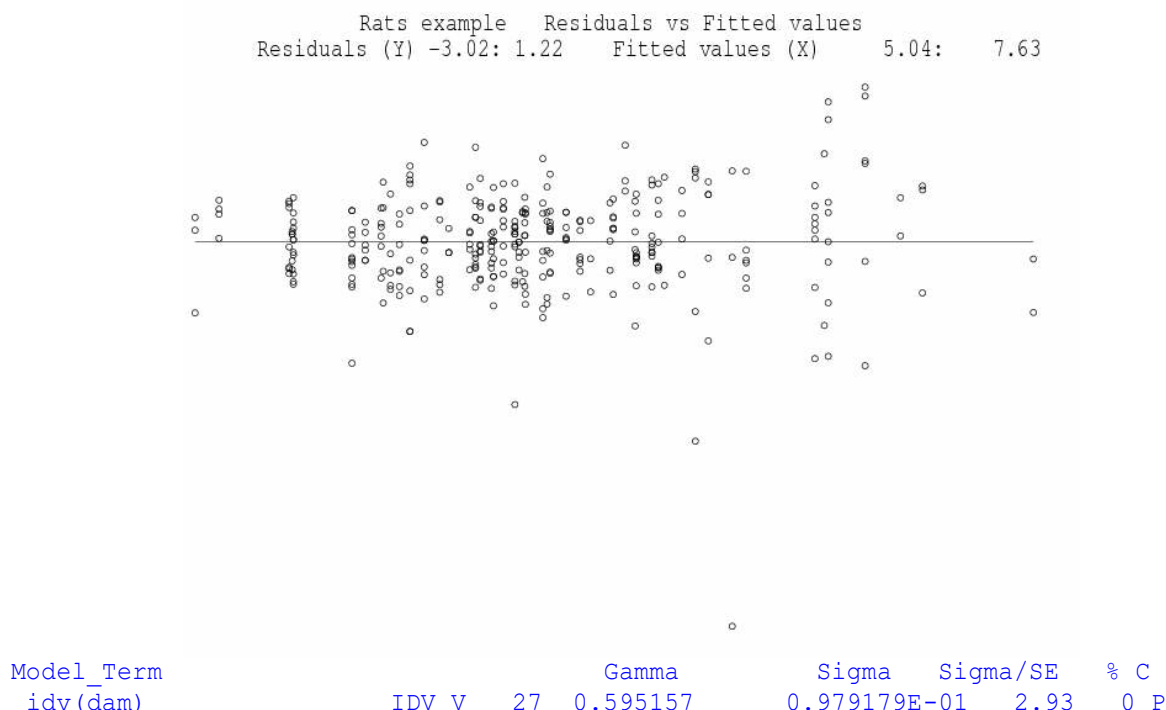


Figure 16.1: Residual plot for the rat data

|                     |           |         |           |         |         |
|---------------------|-----------|---------|-----------|---------|---------|
| units               | SCA_V 322 | 1.00000 | 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     |
| units               | SCA_V 322 | 1.00000 | 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 - Cox & Snell 1981
regulator 8 !A          # regulators numbered within setting stations
setstat 10 !A           # setting stations each set 4-8 regulators
voltage
teststat 4 !=V0 !=1 !MOD 4 !=1 # testing stations tested each regulator
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 follows.

```
1 LogL= 188.604      S2= 0.67074E-01    255 df : 1 components restrained
2 LogL= 197.156      S2= 0.62537E-01    255 df : 1 components restrained
3 LogL= 201.487      S2= 0.56734E-01    255 df : 1 components restrained
4 LogL= 202.998      S2= 0.53039E-01    255 df : 1 components restrained
5 LogL= 203.240      S2= 0.51306E-01    255 df : 1 components restrained

      : idv(setstat.teststat) dropped from model because variance is negligible.
6 LogL= 203.242      S2= 0.51142E-01    255 df    0.6428E-01 0.2334      0.000
0.6017
Final parameter values                0.6428E-01 0.2334      0.000
0.6018

- - - Results from analysis of voltage - - -
Akaike Information Criterion      -398.48 (assuming 4 parameters).
Bayesian Information Criterion    -384.32

Model_Term          IDV_V    4    0.642752E-01    0.328714E-02    0.98    0 P
idv(teststat)
idv(setstat)         IDV_V   10    0.233418      0.119374E-01    1.35    0 P
idv(setstat.teststat) IDV_V   40    0.00000      0.00000      0.00    0 B
idv(setstat.regulator) IDV_V   80    0.601817     0.307779E-01    3.64    0 P
units                SCA_V  256    1.00000     0.511416E-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 have 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 zero or a small positive value (`!NOZERO`) for any subsequent iterations 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.

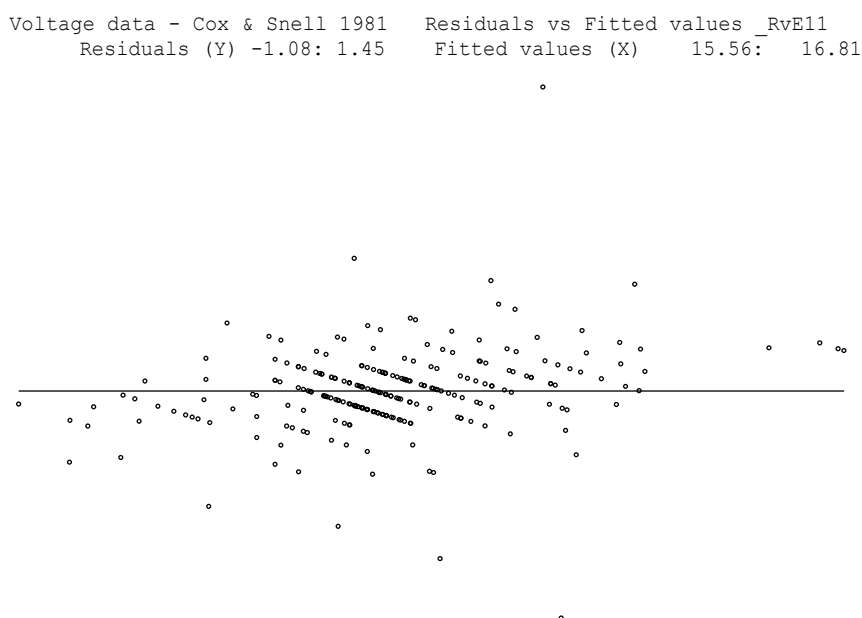


Figure 16.2: Residual plot for the voltage data

The REML log-likelihood from the model without the `setstat.teststat` term was 203.242, basically the same as the REML log-likelihood for the previous model.

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

Table 16.3: REML log-likelihood ratio for the variance components in the voltage data

| terms               | REML log-likelihood | $-2 \times \text{difference}$ | P-value |
|---------------------|---------------------|-------------------------------|---------|
| – setstat           | 200.31              | 5.864                         | 0.0077  |
| – setstat.regulator | 184.15              | 38.19                         | 0.0000  |
| – teststat          | 199.71              | 7.064                         | 0.0039  |

## 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.

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

```
Multivariate grass height data
  tmt 2 !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 Trait.tmt !r idv(units)
residual idv(units.Trait)
```

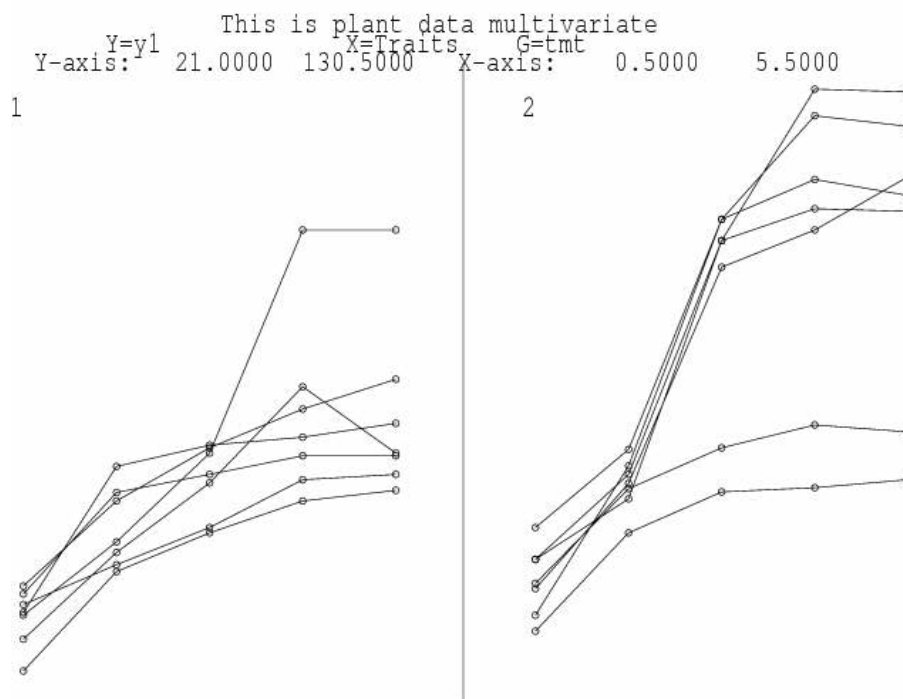


Figure 16.3: Trellis plot of the height for each of 14 plants

The focus is modelling of the error variance for the data. Specifically, we fit the multivariate regression model given by

$$\mathbf{Y} = \mathbf{DT} + \mathbf{E} \quad (16.1)$$

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

$$\text{var}(\text{vec}(\mathbf{E})) = \mathbf{I}_{14} \otimes \mathbf{\Sigma} \quad (16.2)$$

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

The variance models used for  $\mathbf{\Sigma}$  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 would only allow us to fit the final (UnStructured) variance model.

Table 16.4: Summary of variance models fitted to the plant data

| model                  | number of parameters | REML log-likelihood | BIC    |
|------------------------|----------------------|---------------------|--------|
| uniform                | 2                    | -196.88             | 401.94 |
| power                  | 2                    | -182.98             | 374.15 |
| heterogeneous power    | 6                    | -171.50             | 367.56 |
| antependence (order 1) | 9                    | -160.37             | 357.59 |
| unstructured           | 15                   | -158.04             | 377.49 |

The split plot in time model can be fitted in two ways, either by fitting a `units` term plus an independent residual as above, or by specifying a `CORU` variance model for the  $R$ -structure as follows

```
y1 y3 y5 y7 y10 ~ Trait tmt Trait.tmt
residual id(units).coru(Trait)
```

The two forms for  $\Sigma$  are given by

$$\begin{aligned}\Sigma &= \sigma_1^2 \mathbf{J} + \sigma_2^2 \mathbf{I}, & \text{units} \\ \Sigma &= \sigma_e^2 \mathbf{I} + \sigma_e^2 \rho (\mathbf{J} - \mathbf{I}), & \text{CORU}\end{aligned}\quad (16.3)$$

It follows that

$$\begin{aligned}\sigma_e^2 &= \sigma_1^2 + \sigma_2^2 \\ \rho &= \frac{\sigma_1^2}{\sigma_1^2 + \sigma_2^2}\end{aligned}\quad (16.4)$$

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 the following  $\sigma_e^2 = 286.310 \approx 159.816 + 126.494 = 286.310$ ;  $159.816/286.310 = 0.5582$ .

```
#
# !ridv(units.Trait)
#
1 LogL= -204.593      S2=  224.61      60 df
2 LogL= -201.357      S2=  187.89      60 df
3 LogL= -198.545      S2=  156.20      60 df
4 LogL= -197.241      S2=  138.02      60 df
5 LogL= -196.893      S2=  128.65      60 df
6 LogL= -196.877      S2=  126.62      60 df
7 LogL= -196.877      S2=  126.49      60 df
8 LogL= -196.877      S2=  126.49      60 df

- - - Results from analysis of y1 y3 y5 y7 y10 - - -
Akaike Information Criterion      397.75 (assuming 2 parameters).
Bayesian Information Criterion    401.94

Approximate stratum variance decomposition
Stratum      Degrees-Freedom      Variance      Component Coefficients
idv(units)      12.00      925.573      5.0      1.0
Residual Variance      48.00      126.494      0.0      1.0

Model_Term      IDV_V      14      1.26342      159.816      2.11      0 P
units.Trait      SCA_V      70      1.00000      126.494      4.90      0 P
```

## 16.5 Balanced repeated measures - Height

```
#
# id(units).coru(Trait)
#
1 LogL= -198.873      S2= 228.71      60 df
2 LogL= -197.950      S2= 234.83      60 df
3 LogL= -197.151      S2= 252.79      60 df
4 LogL= -196.878      S2= 284.06      60 df
5 LogL= -196.877      S2= 286.30      60 df
6 LogL= -196.877      S2= 286.31      60 df

- - - Results from analysis of y1 y3 y5 y7 y10 - - -
Akaike Information Criterion      397.75 (assuming 2 parameters).
Bayesian Information Criterion    401.94

Model_Term                      Gamma      Sigma      Sigma/SE      % C
idv(units).coru(Trait)          70 effects
units.Trait                     SCA_V    70    1.00000      286.310      3.65      0 P
Trait                          COR_R    5    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 in weeks. The coding for this is

```
y1 y3 y5 y7 y10 ~ Trait tmt Trait.tmt
residual idv(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
2 LogL= -183.773      S2= 440.42      60 df
3 LogL= -183.070      S2= 337.51      60 df
4 LogL= -182.981      S2= 297.16      60 df
5 LogL= -182.979      S2= 302.31      60 df
6 LogL= -182.979      S2= 301.45      60 df

- - - 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
idv(units).exp(Trait)          70 effects
units.Trait                     SCA_V    70    1.00000      301.449      3.12      0 P
Trait                          EXP_P    5    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  $\phi$  too close to 1. For example, use of days in this example would result in an estimate for  $\phi$  of about 0.993.

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  $\Sigma$  by

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

where  $D$  is a diagonal matrix of variances and  $C$  is a correlation matrix with elements given by



## 16.5 Balanced repeated measures - Height

$c_{ij} = \phi^{|t_i - t_j|}$ . The coding for this is

```
y1 y3 y5 y7 y10 ~ Trait tmt Trait.tmt
residual id(units).exph(Trait !INIT 0.5 100 200 300 300 300 !COORD 1 3 5 7 10)
```

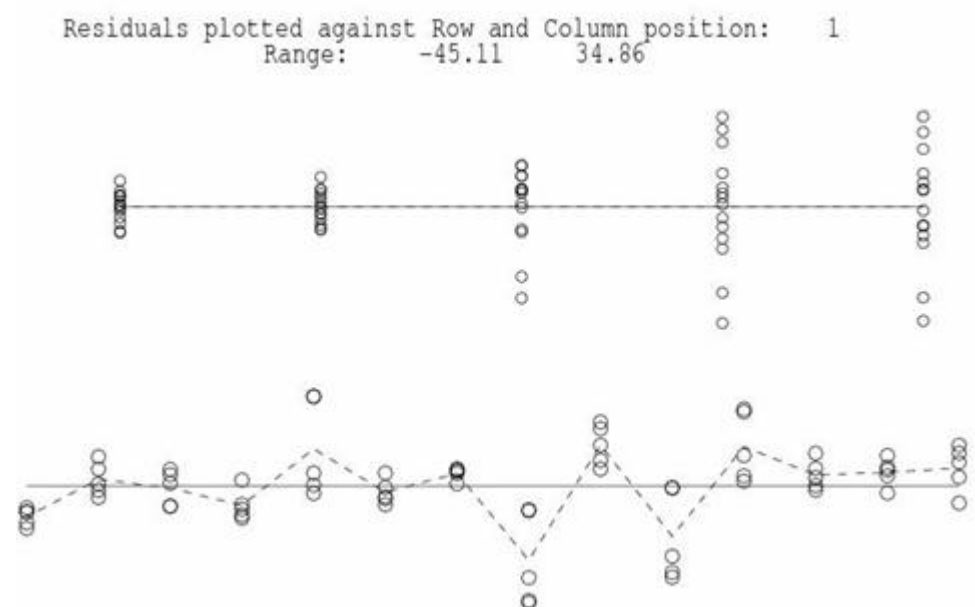


Figure 16.4: Residual plots for the EXP variance model for the plant data

Abbreviated output from this analysis is

```
:
 7 LogL= -171.575      S2=  1.0000      60 df
 8 LogL= -171.516      S2=  1.0000      60 df
 9 LogL= -171.501      S2=  1.0000      60 df
10 LogL= -171.497      S2=  1.0000      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      0.907013      0.907013      21.92      0 P
Trait      EXP_V      1      61.0645      61.0645      2.12      0 P
Trait      EXP_V      2      72.8887      72.8887      2.00      0 P
Trait      EXP_V      3      309.590      309.590      2.22      0 P
Trait      EXP_V      4      436.947      436.947      2.51      0 P
Trait      EXP_V      5      382.660      382.660      2.73      0 P
Covariance/Varianace/Correlation Matrix  Residual
61.29      0.8230      0.6773      0.5574      0.4161
54.88      72.55      0.8230      0.6773      0.5057
93.39      123.5      310.2      0.8230      0.6144
91.32      120.7      303.3      438.0      0.7466
63.81      84.36      212.0      306.0      383.6
```

The last two models we fit are the antedependence model of order 1 and the unstructured model.

## 16.5 Balanced repeated measures - Height

Starting values need not actually be supplied in this example (the defaults are adequate) but are supplied to demonstrate the syntax. We use the REML estimate of  $\Sigma$  from the heterogeneous power model shown in the previous output. The antedependence model models  $\Sigma$  by the inverse cholesky decomposition

$$\Sigma^{-1} = \mathbf{U}\mathbf{D}\mathbf{U}'$$

where  $\mathbf{D}$  is a diagonal matrix and  $\mathbf{U}$  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  $\mathbf{D}$  and 4 in  $\mathbf{U}$ .

The input is given by

```
!ASSIGN ANTEI !< !INIT
60.16
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 Trait.tmt
residual id(units).ante(Trait $ANTEI)
```

The abbreviated output file is

```
...
4 LogL= -161.368      S2= 1.0000      60 df
5 LogL= -160.414      S2= 1.0000      60 df
6 LogL= -160.370      S2= 1.0000      60 df
7 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
Note: Akaike parameter count excludes components bound at/near zero and
      is adjusted for any identifiability constraints on factor loadings.

Model_Term                                Sigma      Sigma      Sigma/SE      % C
id(units).ante(Trait)                    70 effects
Trait      ANTE_U 1 1 0.268648E-01      0.268648E-01      2.44      0 P
Trait      ANTE_U 2 1 -0.628409      -0.628409      -2.55      0 P
Trait      ANTE_U 2 2 0.372828E-01      0.372828E-01      2.41      0 P
Trait      ANTE_U 3 2 -1.49098      -1.49098      -2.55      0 P
Trait      ANTE_U 3 3 0.599598E-02      0.599598E-02      2.43      0 P
Trait      ANTE_U 4 3 -1.28034      -1.28034      -6.19      0 P
Trait      ANTE_U 4 4 0.789727E-02      0.789727E-02      2.44      0 P
Trait      ANTE_U 5 4 -0.967812      -0.967812      -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
37.19      0.5945      0.3549      0.3114      0.3040
23.37      41.55      0.5970      0.5239      0.5113
34.82      61.91      258.8      0.8775      0.8565
44.58      79.26      331.4      550.8      0.9761
43.14      76.70      320.7      533.1      541.6
```

The iterative sequence converged and the antedependence parameter estimates are printed column-wise by time, the column of  $\mathbf{U}$  and the element of  $\mathbf{D}$ , *i.e.*

$$D = \text{diag} \begin{bmatrix} 0.0269 \\ 0.0373 \\ 0.0060 \\ 0.0079 \\ 0.0391 \end{bmatrix}, U = \begin{bmatrix} 1 & -0.6284 & 0 & 0 & 0 \\ 0 & 1 & -1.4911 & 0 & 0 \\ 0 & 0 & 1 & -1.2804 & 0 \\ 0 & 0 & 0 & 1 & -0.9678 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Finally, the input and output files for the unstructured model are presented below. The REML estimate of  $\Sigma$  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)
```

with abbreviated output file

```
1 LogL= -160.368      S2=  1.0000      60 df
2 LogL= -159.073      S2=  1.0000      60 df
3 LogL= -158.268      S2=  1.0000      60 df
4 LogL= -158.041      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
Covariance/Variance/Correlation Matrix US Residual
37.22      0.5953      0.5259      0.4942      0.5193
23.40      41.50      0.5972      0.3813      0.4175
51.65      61.92      259.1      0.8778      0.8827
70.79      57.67      331.8      551.3      0.9761
73.77      62.61      330.8      533.6      542.0
```

However, the usual syntax for fitting an unstructured error model for multivariate data is to omit the !ASUV qualifier and write

```

:
grass.asd !SKIP 1
y1 y3 y5 y7 y10 ~ Trait tmt Trait.tmt
residual id(units).us(Trait) # ASReml generates initial values

```

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 plant data**

| model                   | treatment (df = 1) | treatment.time (df = 4) |
|-------------------------|--------------------|-------------------------|
| uniform                 | 9.41               | 5.10                    |
| power                   | 6.88               | 6.11                    |
| heterogenous power      | 0.00               | 4.31                    |
| antedepedence (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

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

where  $\mathbf{\Sigma}_c$  and  $\mathbf{\Sigma}_r$  are  $15 \times 15$  and  $10 \times 10$  matrix functions of the column ( $\phi_c$ ) and row ( $\phi_r$ ) autoregressive parameters respectively. Gilmour *et al.* (1997) recommend revision of the current spatial model based on the use of diagnostics such as the sample variogram of the residuals (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 conjunction 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 - Rep 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 3
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 !DOPART $1

!PART 1 # AR1*AR1
y ~ mu variety
residual arlv(column).ar1(row)
PREDICT var !TWOSTAGEWEIGHTS

!PART 2 # AR1*AR1 + units
y ~ mu variety !r idv(units)
residual arlv(column).ar1(row)
PREDICT var !TWOSTAGEWEIGHTS

!PART 3 # Incomplete Blocks
y ~ mu variety !r idv(Rep) idv(Rowblk) idv(Colblk)
residual idv(units)
PREDICT variety !TWOSTAGEWEIGHTS

```

Abbreviated **ASReml** output file is presented below. The iterative sequence has converged to column and row correlation parameters of (0.68377, 0.45859), respectively. The plot size was 14ft lengthwise by 5ft in width. 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

$$\sigma^2 \mathbf{\Sigma} = \sigma^2 (\mathbf{\Sigma}_c \otimes \mathbf{\Sigma}_r) + \psi \mathbf{I}_{150} \quad (16.6)$$

where  $\psi$  is the ratio of nugget variance to error variance ( $\sigma^2$ ). 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).

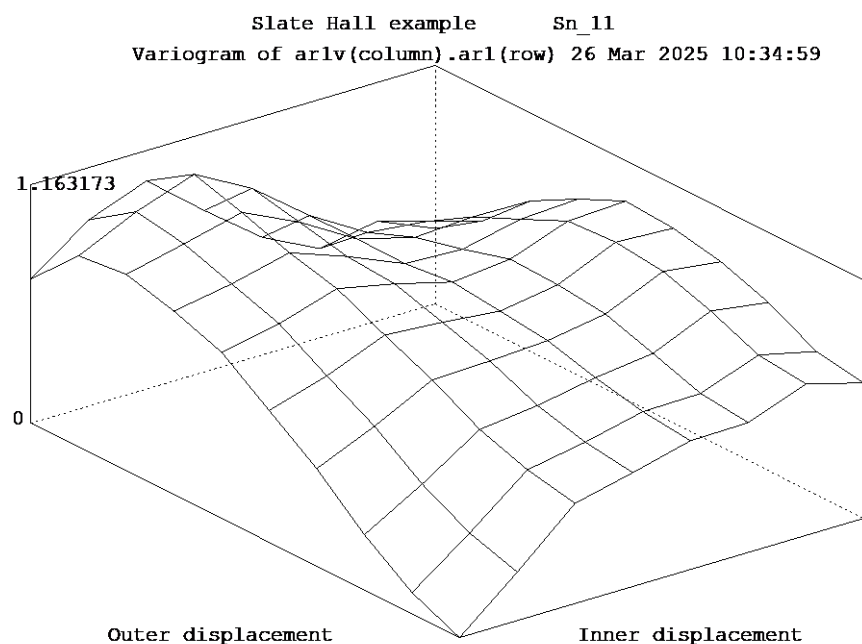


Figure 16.5: Sample variogram of the residuals from the AR1×AR1 model

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                    | 10.21            | 60.5 |
| IB              | -707.79             | 4                    | 8.84             | 62.0 |

```
#
# AR1*AR1
#
1 LogL= -739.681    S2= 36034.    125 df : 1 components restrained
2 LogL= -718.221    S2= 28664.    125 df : 1 components restrained
3 LogL= -704.627    S2= 29007.    125 df
4 LogL= -700.695    S2= 34317.    125 df
5 LogL= -700.326    S2= 38352.    125 df
6 LogL= -700.323    S2= 38718.    125 df
7 LogL= -700.322    S2= 38752.    125 df
```

- - - Results from analysis of yield - - -

```
Akaike Information Criterion    1406.64 (assuming 3 parameters).
Bayesian Information Criterion   1415.13
```

| Model_Term            |             | Gamma    | Sigma    | Sigma/SE | % C |
|-----------------------|-------------|----------|----------|----------|-----|
| arlv(column).arl(row) | 150 effects |          |          |          |     |
| Residual              | SCA_V 150   | 1.00000  | 38752.1  | 5.00     | 0 P |
| column                | AR_R 15     | 0.683769 | 0.683769 | 10.80    | 0 P |
| row                   | AR_R 10     | 0.458583 | 0.458583 | 5.55     | 0 P |

| Source of Variation | NumDF | DenDF | F-inc  | P-inc |
|---------------------|-------|-------|--------|-------|
| 8 mu                | 1     | 12.8  | 851.03 | <.001 |
| 6 variety           | 24    | 80.0  | 13.04  | <.001 |

## 16.6 Spatial analysis of a field experiment – Barley

```
#
# AR1*AR1 + units
#
1 LogL= -740.735      S2= 33225.      125 df : 1 components restrained
2 LogL= -727.521      S2= 31195.      125 df : 3 components restrained
3 LogL= -710.639      S2= 23844.      125 df : 2 components restrained
4 LogL= -699.234      S2= 33979.      125 df
5 LogL= -697.471      S2= 35667.      125 df
6 LogL= -696.893      S2= 46074.      125 df
7 LogL= -696.826      S2= 44853.      125 df
8 LogL= -696.823      S2= 45708.      125 df
9 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 | 0.106156 | 4861.50  | 2.72     | 0 P |
| arlv(column).arl(row) |       | 150 | effects  |          |          |     |
| Residual              | SCA_V | 150 | 1.00000  | 45795.7  | 2.74     | 0 P |
| column                | AR_R  | 15  | 0.843796 | 0.843796 | 12.33    | 0 P |
| row                   | AR_R  | 10  | 0.682687 | 0.682687 | 6.68     | 0 P |

| Source of Variation | Wald F statistics |       |        | P-inc |
|---------------------|-------------------|-------|--------|-------|
|                     | NumDF             | DenDF | F-inc  |       |
| 8 mu                | 1                 | 3.5   | 259.85 | <.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.

```
#
# Incomplete Blocks
#
1 LogL= -734.184      S2= 26778.      125 df  0.1000      0.1000      0.1000
2 LogL= -720.066      S2= 16595.      125 df  0.1477      0.3797      0.3765
3 LogL= -711.122      S2= 11176.      125 df  0.2391      0.8902      0.8756
4 LogL= -708.253      S2= 8997.7      125 df  0.3622      1.459       1.418
5 LogL= -707.791      S2= 8149.6      125 df  0.4890      1.878       1.796
6 LogL= -707.786      S2= 8062.4      125 df  0.5268      1.934       1.837
7 LogL= -707.786      S2= 8061.8      125 df  0.5287      1.934       1.837
Final parameter values                                0.5287      1.934      1.837
```

```
- - - Results from analysis of yield - - -
Akaike Information Criterion 1423.57 (assuming 4 parameters).
Bayesian Information Criterion 1434.88
```

| Approximate stratum variance decomposition |                 |          |           |              |     |     |
|--|-----------------|----------|-----------|--------------|-----|-----|
| Stratum                                    | Degrees-Freedom | Variance | Component | Coefficients |     |     |
| idv(Rep)                                   | 5.00            | 266670.  | 25.0      | 5.0          | 5.0 | 1.0 |
| idv(Rowblk)                                | 24.00           | 74887.5  | 0.0       | 4.3          | 0.0 | 1.0 |
| idv(Colblk)                                | 23.66           | 71352.5  | 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 |



## 16.6 Spatial analysis of a field experiment – Barley

|             |       |     |         |         |      |     |
|-------------|-------|-----|---------|---------|------|-----|
| idv(Rowblk) | IDV_V | 30  | 1.93444 | 15595.1 | 3.06 | 0 P |
| idv(Colblk) | IDV_V | 30  | 1.83725 | 14811.5 | 3.04 | 0 P |
| units       | SCA_V | 150 | 1.00000 | 8061.81 | 6.01 | 0 P |

| Source of Variation | Wald F statistics |       |         |       |
|---------------------|-------------------|-------|---------|-------|
|                     | NumDF             | DenDF | F-inc   | P-inc |
| 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.

```
#
# AR1*AR1
#
variety      Predicted_Value Standard_Error Ecode
1            1257.9747         64.6126 E
2            1501.4509         64.9762 E
3            1404.9878         64.6239 E
4            1412.5671         64.9006 E
5            1514.4748         65.5869 E
:
23           1311.4795         64.0708 E
24           1586.7744         64.6982 E
25           1592.0173         63.5880 E
SED: Standard Error of Difference: Min    52.4863    Mean    59.0142    Max    64.1528
```

```
#
# AR1*AR1 + units
#
variety      Predicted_Value Standard_Error Ecode
1            1245.5875         97.8557 E
2            1516.2324         97.8439 E
3            1403.9881         98.2364 E
4            1404.9230         97.9841 E
5            1471.6259         98.3573 E
:
23           1316.8752         98.0368 E
24           1557.5321         98.1238 E
25           1573.8949         97.9768 E
SED: Standard Error of Difference: Min    56.9155    Mean    60.4961    Max    63.8238
```

```
#
# Incomplete Blocks
#
The ignored set: Rep Rowblk Colblk
variety      Predicted_Value Standard_Error Ecode
1            1283.5870         60.1994 E
2            1549.0133         60.1994 E
3            1420.9307         60.1994 E
4            1451.8554         60.1994 E
5            1533.2749         60.1994 E
:
```

## 16.7 Unreplicated early generation variety trial – Wheat

|  |           |           |         |     |         |  |
|--|-----------|-----------|---------|-----|---------|--|
| 23                                     | 1329.1089 | 60.1994 E |         |     |         |  |
| 24                                     | 1546.4699 | 60.1994 E |         |     |         |  |
| 25                                     | 1630.6285 | 60.1994 E |         |     |         |  |
| SED: Standard Error of Difference: Min | 62.0193   | Mean      | 62.0193 | Max | 62.0193 |  |

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 a few lines for each model below.

```
#
# AR1*AR1
#
Predicted values with Effective Replication (ER)
Divide ER by variance ( 38752.08 ) to get weights.
Heron: 1 1257.97 22.1479
Heron: 2 1501.45 20.6811
Heron: 3 1404.99 22.5263
:
Heron: 23 1311.48 23.2597
Heron: 24 1586.77 24.6052
Heron: 25 1592.02 26.0960

#
# AR1*AR1 + units
#
Predicted values with Effective Replication (ER)
Divide ER by variance ( 45795.66 ) to get weights.
Heron: 1 1245.59 23.8854
Heron: 2 1516.23 22.4431
Heron: 3 1403.99 24.1940
:
Heron: 23 1316.88 24.8336
Heron: 24 1557.53 25.8631
Heron: 25 1573.89 26.0521

#
# Incomplete Blocks
#
Predicted values with Effective Replication (ER)
Divide ER by variance ( 8061.805 ) to get weights.
Heron: 1 1283.59 4.03145
Heron: 2 1549.01 4.03145
Heron: 3 1420.93 4.03145
:
Heron: 23 1329.11 4.03145
Heron: 24 1546.47 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 Rolblk effects. The precision from the spatial analyses ( $45796.58/23.8842 = 1917.442$  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.

## 16.7 Unreplicated early generation variety trial – Wheat

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

```
Tullibigeal trial
  linenum *
  yield
  weed
  column 10
  row 67
  variety 532 # test lines 1:525, check lines 526:532
wheat.asd !SKIP 1 !EXTRA 3 !DOPART $1

!PART 1 # AR1 x I
y ~ mu weed mv !r idv(variety)
residual ar1v(row).id(column)

!PART 2 # AR1 x AR1
y ~ mu weed mv !r idv(variety)
residual ar1v(row).ar1(column)

!PART 3 # AR1 x AR1 + column trend
y ~ mu weed pol(column,-1) mv !r idv(variety)
residual ar1v(row).ar1(column)

!PART 4 # AR1 x AR1 + column trend + nugget
y ~ mu weed pol(column,-1) mv !r idv(variety) idv(units)
residual ar1v(row).ar1(column)
PREDICT variety column 5.5
```

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 (part 1), which assumes the variance model for the plot effects within columns is described by a first order autoregressive process. The abbreviated output file is

```
:
```

|    |                |            |        |
|----|----------------|------------|--------|
| 10 | LogL= -4239.88 | S2= 86275. | 666 df |
| 11 | LogL= -4239.88 | S2= 86290. | 666 df |
| 12 | LogL= -4239.88 | S2= 86297. | 666 df |
| 13 | LogL= -4239.88 | S2= 86300. | 666 df |

## 16.7 Unreplicated early generation variety trial – Wheat

```

- - - Results from analysis of yield - - -
Akaike Information Criterion      8485.76 (assuming 3 parameters).
Bayesian Information Criterion    8499.26

Model_Term              IDV_V  532  0.959695      Sigma  Sigma/SE  % C
idv( variety)              IDV_V  532  0.959695      82822.7    8.99    0 P
arlv( row).id( column)      670 effects
Residual                  SCA_V  670  1.00000      86301.1    9.12    0 P
row                        AR_R   67  0.672551      0.672551   16.09    0 P

                                Wald F statistics
Source of Variation      NumDF  DenDF  F-inc      P-inc
7 mu                      1      83.2  9768.32    <.001
3 weed                    1      477.1  109.17     <.001
Note: The DenDF values are calculated ignoring fixed/boundary/singular

```

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 (part 2) is

```

:
10 LogL= -4233.65      S2=  83094.      666 df
11 LogL= -4233.65      S2=  83111.      666 df
12 LogL= -4233.65      S2=  83118.      666 df

- - - Results from analysis of yield - - -
Akaike Information Criterion      8475.29 (assuming 4 parameters).
Bayesian Information Criterion    8493.30

Model_Term              IDV_V  532  1.06058      Sigma  Sigma/SE  % C
idv( variety)              IDV_V  532  1.06058      88156.3    9.92    0 P
arlv( row).ar1( column)    670 effects
Residual                  SCA_V  670  1.00000      83120.9    8.89    0 P
row                        AR_R   67  0.685621      0.685621   16.68    0 P
column                    AR_R   10  0.286085      0.286085    3.87    0 P

                                Wald F statistics
Source of Variation      NumDF  DenDF  F-inc      P-inc
7 mu                      1      41.6  6233.81    <.001
3 weed                    1      491.2  85.75     <.001

```

The change in **REML** log-likelihood is significant ( $\chi^2_1 = 12.46, p < 0.001$ ) 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 (see part 3). The linear regression of column number on yield is significant ( $t = -2.95$ ). 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.

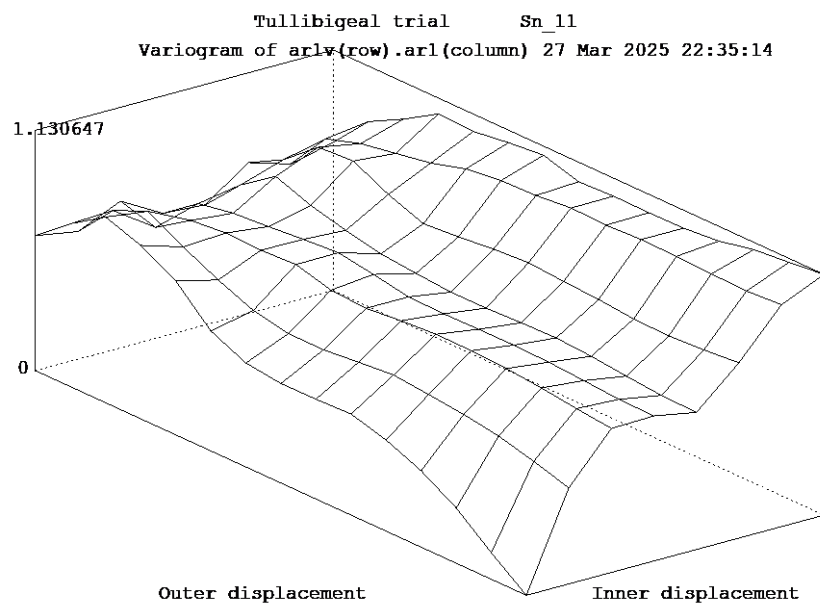


Figure 16.6: Sample variogram of the residuals from the  $AR1 \times AR1$  model for the Tullibigeal data

The abbreviated output for this model and the final model in which a nugget effect has been included is

```
# AR1 x AR1 + column trend
:
10 LogL= -4225.63      S2=  77811.      665 df
11 LogL= -4225.63      S2=  77813.      665 df
12 LogL= -4225.63      S2=  77814.      665 df
```

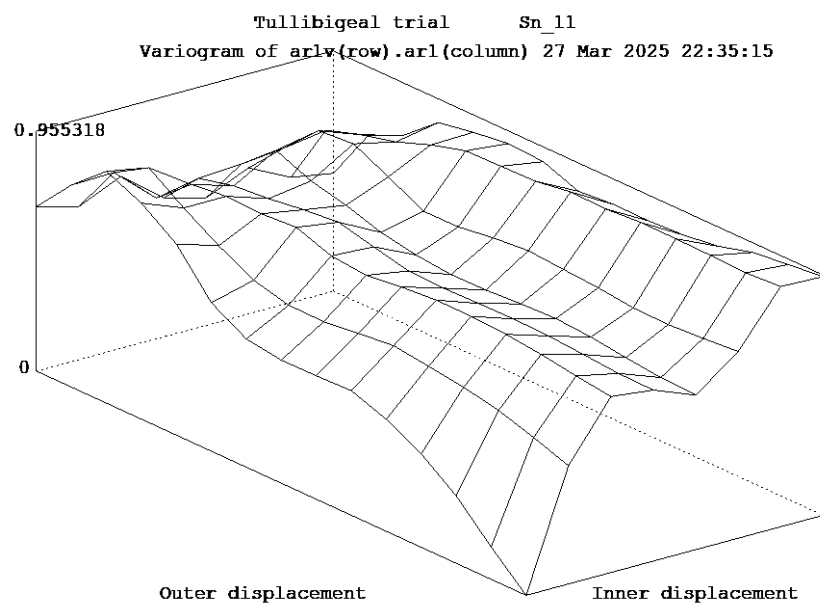


Figure 16.7: Sample variogram of the residuals from the  $AR1 \times AR1 + \text{pol}(\text{column}, -1)$  model for the Tullibigeal data

## 16.7 Unreplicated early generation variety trial – Wheat

- - - Results from analysis of yield - - -

Akaike Information Criterion 8459.26 (assuming 4 parameters).  
 Bayesian Information Criterion 8477.26  
 Coefficient of Determination: NDF 2.00 DenDF 100.75 Fall 50.31 CD 49.97

| Model_Term            |       |     | Gamma    | Sigma    | Sigma/SE | % C |
|-----------------------|-------|-----|----------|----------|----------|-----|
| idv(variety)          | IDV_V | 532 | 1.14385  | 89007.7  | 9.92     | 0 P |
| arlv(row).arl(column) |       | 670 | effects  |          |          |     |
| Residual              | SCA_V | 670 | 1.00000  | 77814.4  | 8.78     | 0 P |
| row                   | AR_R  | 67  | 0.671570 | 0.671570 | 15.67    | 0 P |
| column                | AR_R  | 10  | 0.266183 | 0.266183 | 3.53     | 0 P |

| Source of Variation | Wald F statistics |       |         | P-inc |
|---------------------|-------------------|-------|---------|-------|
|                     | NumDF             | DenDF | F-inc   |       |
| 7 mu                | 1                 | 42.5  | 7070.14 | <.001 |
| 3 weed              | 1                 | 457.4 | 91.89   | <.001 |
| 8 pol(column,-1)    | 1                 | 50.8  | 8.73    | 0.005 |

# AR1 x AR1 + column trend + nugget

```

:
9 LogL= -4220.26      S2=  54717.      665 df
10 LogL= -4220.26     S2=  54715.      665 df
11 LogL= -4220.26     S2=  54715.      665 df

```

- - - Results from analysis of yield - - -

Akaike Information Criterion 8450.52 (assuming 5 parameters).  
 Bayesian Information Criterion 8473.02  
 Coefficient of Determination: NDF 2.00 DenDF 41.64 Fall 45.62 CD 68.66

| Model_Term            |       |     | Gamma    | Sigma    | Sigma/SE | % C |
|-----------------------|-------|-----|----------|----------|----------|-----|
| idv(variety)          | IDV_V | 532 | 1.34824  | 73768.7  | 7.08     | 0 P |
| idv(units)            | IDV_V | 670 | 0.556402 | 30443.5  | 3.77     | 0 P |
| arlv(row).arl(column) |       | 670 | effects  |          |          |     |
| Residual              | SCA_V | 670 | 1.00000  | 54714.9  | 5.15     | 0 P |
| row                   | AR_R  | 67  | 0.837503 | 0.837503 | 18.67    | 0 P |
| column                | AR_R  | 10  | 0.375382 | 0.375382 | 3.26     | 0 P |

| Source of Variation | Wald F statistics |       |         | P-inc |
|---------------------|-------------------|-------|---------|-------|
|                     | NumDF             | DenDF | F-inc   |       |
| 7 mu                | 1                 | 13.6  | 4241.34 | <.001 |
| 3 weed              | 1                 | 469.0 | 86.39   | <.001 |
| 8 pol(column,-1)    | 1                 | 18.5  | 4.84    | 0.041 |

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
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
column is evaluated at      5.5000

```

## 16.8 Multi-Environment Trials

Model terms involving weed are predicted at the average: 0.4597

| variety                                   | Predicted_Value | Standard_Error | Ecode |
|---|-----------------|----------------|-------|
| 1   | 2915.7122       | 179.3117       | E     |
| 2   | 2956.2749       | 178.7802       | E     |
| 3   | 2871.2965       | 176.9942       | E     |
| 4   | 2985.0071       | 178.7451       | E     |
| 5   | 2776.8093       | 179.3483       | E     |
| ...                                       |                 |                |       |
| 523                                       | 2903.4762       | 179.5439       | E     |
| 524                                       | 2738.5678       | 178.8372       | E     |
| 525                                       | 2668.4903       | 179.2244       | E     |
| 526                                       | 2384.5149       | 44.2183        | E     |
| 527                                       | 2695.6017       | 133.4577       | E     |
| 528                                       | 2725.5665       | 112.2614       | E     |
| 529                                       | 2698.3584       | 103.9252       | E     |
| 530                                       | 3008.9251       | 112.3068       | E     |
| 531                                       | 3018.6062       | 112.2712       | E     |
| 532                                       | 3065.9822       | 112.6675       | E     |
| SED: Overall Standard Error of Difference |                 |                | 245.8 |

Note that the (replicated) check lines have lower **SE** than the (unreplicated) test lines. There will also be large differences in **SEDs**. Rather than obtaining the large table of all **SEDs**, you could do the prediction in parts

```
PREDICT variety 1:525 column 5.5
PREDICT variety 526:532 column 5.5 !SED
```

to examine the matrix of pairwise prediction errors of variety differences.

## 16.8 Multi-Environment Trials

In this section we explore some models that can be fitted to multi-environment trials, in particular showing the application of factor analytic models. A multi-environment trial is a series of experiments with treatments in common. Our examples are drawn from field crops where the aim is to identify genotypes which consistently yield well over a region in several seasons.

The traditional approach involved analysing the experiments separately, saving the means, and then performing some weighted analysis of the means. The methods described in this section build on spatial analysis, incorporate trial accuracy and allow for unbalance in genotype representation. For further reading, see Smith *et al.* (2001, 2005).

Multi-environment trials have three basic forms. Early generation trials have few sites and few replicates but many lines to compare. Current late-stage trials have more 3-4 replicates, 10-15 sites and 30-50 entries of elite lines grown at most sites. National analyses draw together results from multiple late-stage trials using results from several years and regions, many sites and many lines. We will consider these three cases to discuss some features of **ASReml**. These analyses should not be considered as definitive, typical or recommended as they demonstrate just one approach. They are chosen to demonstrate syntactical issues, not statistical issues.

### 16.8.1 An early generation trial

Early-stage multi-environment trials typically have many genotypes but limited seed. Consequently, within site replication of test lines is low (1 or 2) so that lines can be tested at more

locations. Traditionally, grid plot designs have been used where standard/reference/check lines are highly replicated using a systematic grid but partial replicate designs are strongly advocated (Cullis *et al.* 2006).

### Three sites MET example

This example involves 6 check lines and 330 test lines grown at three locations. The first 2 locations were laid out in a  $12 \times 34$  arrangement, 12 replicates of each check line and 1 of each test line. The extra 6 plots were sown to a fill-in variety which was also harvested. The third location was laid out in a  $15 \times 28$  arrangement and had 15 replicates of each check line. The data file `met.dat` is sorted row within col within site. The distributed data file has extra fields we will ignore. The code for an initial combined analysis fitting a common random test effect follows. This code incorporates the results from several preliminary runs involving separate spatial analyses of each site ignoring test. These runs suggested a random row term was required for site 3, col terms for all sites, AR1 was unnecessary for the col dimension of the R structure for site 3, and provided the initial values inserted in the code. The `!SECTION site` qualifier allows **ASReml** to check that the factor `site` does indeed correspond to the 3 sections in the R structure.

```
Early Generation Multi-Environment Trial
seq      # Sequential id
col 15    # Actually 12, 12 and 15 for the sites respectively
row 34    # Actually 34, 34 and 28 for the sites respectively
chk 7     # 0 is fill-in, 1-6 are check lines, 7 is test line
test 336  # 0 is fill-in or check, 7-336 is test line
geno 337  # 1-6 are checks, 7-336 are test lines, 337 is fill-in
yld !*.01
site 3    # Coded 1,2,3
met.dat !SECTION site!EXTRA 6
yld ~ site chk.site,
!r at(site,3).idv(row !INIT 0.02) at(site,1).idv(col !INIT 0.90),
at(site,2).idv(col !INIT 0.40) at(site,3).idv(col !INIT 0.036) idv(test)
residual at(site,1,2).ar1(col).arlv(row) at(site,3).id(col).arlv(row)
PREDICT chk site
PREDICT test site chk 7
```

The LogL from this run is -314.262 and the parameter estimates follow

```
- - - Results from analysis of yld - - -
Akaike Information Criterion      654.52 (assuming 13 parameters).
Bayesian Information Criterion    720.83
```

| Model_Term                    |       |     | Sigma        | Sigma        | Sigma/SE | % C |
|-------------------------------|-------|-----|--------------|--------------|----------|-----|
| idv(test)                     | IDV_V | 336 | 0.103070     | 0.103070     | 7.02     | 0 P |
| at(site,1).ar1(col).arlv(row) |       | 408 | effects      |              |          |     |
| col                           | AR_R  | 12  | 0.195917     | 0.195917     | 3.65     | 0 P |
| row                           | AR_R  | 1   | 0.650403     | 0.650403     | 16.79    | 0 P |
| row                           | AR_V  | 1   | 2.77137      | 2.77137      | 8.82     | 0 P |
| at(site,2).ar1(col).arlv(row) |       | 408 | effects      |              |          |     |
| col                           | AR_R  | 12  | 0.286847     | 0.286847     | 5.47     | 0 P |
| row                           | AR_R  | 1   | 0.574458     | 0.574458     | 13.65    | 0 P |
| row                           | AR_V  | 1   | 0.992641     | 0.992641     | 9.20     | 0 P |
| at(site,3).id(col).arlv(row)  |       | 420 | effects      |              |          |     |
| row                           | AR_R  | 1   | 0.639359     | 0.639359     | 10.11    | 0 P |
| row                           | AR_V  | 1   | 0.120461     | 0.120461     | 6.43     | 0 P |
| at(site,1).idv(col)           | ID_V  | 15  | 0.622585     | 0.622585     | 1.45     | 0 P |
| at(site,2).idv(col)           | ID_V  | 15  | 0.158986     | 0.158986     | 1.40     | 0 P |
| at(site,3).idv(col)           | ID_V  | 15  | 0.483267E-01 | 0.483267E-01 | 1.85     | 0 P |



## 16.8 Multi-Environment Trials

```
at(site,3).idv(row)      ID_V   34  0.235028E-01  0.235028E-01  2.77  0 P
```

|                     |       | Wald F statistics |         |  |
|---------------------|-------|-------------------|---------|--|
| Source of Variation | NumDF |                   | F-inc   |  |
| 8 site              | 3     |                   | 1230.53 |  |
| 9 chk.site          | 20    |                   | 11.36   |  |

Practitioners have taken two views on whether check or control lines should be fitted as fixed effects, or just treated as random effects in the set of genotypes. It may effect the variance of the test/genotype effects. Comparison of test lines with check lines is easier if all are in the same random factor but this analysis takes the former approach. Notice in passing that `chk.site` has 20 rather than 18 degrees of freedom because of the fill-in variety at the first 2 sites.

Our aim in this analysis is to get good predictions of test line effects. To do this we can compare several models for the genetic variances and covariances of test lines across sites. We are interested in the common effect of genotype across sites, but also to know to what extent individual sites diverge from the common genotype rankings. Our naive first model simply fits a common genetic effect, primarily to show why this model is inadequate. It implies the genetic variance is the same at all sites and the genetic correlation between sites is 1. Both assumptions are unlikely given that the residual variances range from 0.12 to 2.77.

Adding `idv(site.test)` to the random model allows for a common covariance less than the common variance. It increases the LogL from -314.262 to -310.879, highly significant and gives component

```
idv(test)      IDV_V   336  0.867588E-01  0.867588E-01  5.46  0 P
site.test      IDV_V  1008  0.440525E-01  0.440525E-01  2.62  0 P
```

This represents a genetic correlation of  $0.663 = 0.0868 / (0.0868 + 0.0441)$  and an average genetic variance of  $0.1308 = 0.08676 + 0.04405$ , up from 0.1031. However, the residual site variances are quite different (20 fold) so it is likely that the genetic variances differ between sites.

Our next model, therefore, fits a common correlation but heterogeneous variances. We drop test from the list of model terms and put the **CORUH** structure on the site component of `site.test`. The **ASReml** code is

```
yld ~ site chk.site,
!r at(site,3).idv(row !INIT 0.02) at(site,1).idv(col !INIT 0.90),
at(site,2).idv(col !INIT 0.40) at(site,3).idv(col !INIT 0.036) coruh(site).id(test)
residual at(site,1,2).ar1(col).arlv(row) at(site,3).id(col).arlv(row)
```

The LogL increases 22.4 ( $P < 0.01$ ) to -288.484 ( $P < 0.01$ ) and the components are

|                      |       | 1008 effects |          |          |      |     |
|----------------------|-------|--------------|----------|----------|------|-----|
| coruh(site).id(test) |       |              |          |          |      |     |
| site                 | COR_R | 1            | 0.418620 | 0.418620 | 6.36 | 0 P |
| site                 | COR_V | 1            | 0.991328 | 0.991328 | 7.95 | 0 P |
| site                 | COR_V | 2            | 0.152324 | 0.152324 | 2.73 | 0 P |
| site                 | COR_V | 3            | 0.122783 | 0.122783 | 7.19 | 0 P |

So, the common correlation under this model is 0.42 (down from 0.66) and the site variances are 0.99, 0.15 and 0.12 respectively; site 1 being particularly high. This **CORUH** model would typically be the first model fitted. In this case there is a different genetic variance at each site but sometimes that will not be the case.

Is the assumption of common correlation justified? Finally, we fit an unstructured model that might be more difficult to converge, here, starting values and additional iterations can assist. This is often equivalent to an XFA1 model for 3 sites but with more sites, the XFA1 model might be more parsimonious. The ASReml code is

```
yld ~ site chk.site,  
!r at(site,3).idv(row !INIT 0.02) at(site,1).idv(col !INIT 0.90),  
at(site,2).idv(col !INIT 0.40) at(site,3).idv(col !INIT 0.036) us(site).id(test)  
residual at(site,1,2).ar1(col).ar1v(row) at(site,3).id(col).ar1v(row)
```

Fitting this model, the Log-Likelihood is increased 1.66 to -286.82, which is not significant ( $P > 0.05$ ) for 2 parameters.

The fitted genetic variance matrix from the unstructured model is

$$\begin{pmatrix} 0.992 & 0.158 & 0.132 \\ 0.158 & 0.073 & 0.078 \\ 0.132 & 0.078 & 0.122 \end{pmatrix}$$

What next? Predicted values for the lines at each site are given in the .pvs file by the PREDICT statement

```
PREDICT site test check 7 # Check 7 is the mean effect for test
```

### Five sites MET example

This second example is also an early generation trial but with more sites (5). It demonstrates a set of five models typically fitted to current late-stage trials. There were 330 genotypes replicated twice at each site except 7 plots were sown to a fill in variety. Notice there is no separate coding of check-plot genotypes in this example.

```
!RENAME !WORK 4 !NO !ARG 1 2 3 4 5 !CONTINUE  
Analysis met.a307  
Plot *  
Block *  
Entry *  
Column *  
Row *  
Genotype !A  
Site !A 5  
Year !I  
Environment !A  
yield !*0.001  
met307.csv !SKIP 1 !EXTRA 5 !DOPART $1  
  
!PART 1  
yield ~ mu Site at(Site).lin(Row) at(Site).lin(Column) mv,  
!r at(Site).Row at(Site).Column at(Site).Block coruh(Site).Geno  
  
!PART 2 3 4 5  
yield ~ mu Site at(Site,1,2,4,5).lin(Row) at(Site,5).lin(Column) mv,  
!r at(Site,1,2,3,5).idv(Row) at(Site).idv(Column) at(Site,2).idv(Block 0.05),  
  
!PART 2// coruh(Site).Genotype  
!PART 3// xfa1(Site).Genotype
```

```
!PART 4// xfa2(Site).Genotype
!PART 5// us(Site).Genotype

!PART 0
residual at(Site).ar1(Row).arlv(Col)
```

We have used !CONTINUE so that each part will take initial values as best it can from the previous us. Part 1 ended after 7 iterations with a LogL of 2563.7 having converged. Part 2 (starting from Part 1 estimates) converged in 11 iterations to LogL 2586.7. These LogL values are not comparable because some fixed effects were dropped when fitting part 2. Both these fitted the genetic variance matrix assuming equal correlation among sites but different variances (CORUH).

Factor Analytic models provide a parsimonious approach generalising the covariance structure. Replacing CORUH with XFA1 in part 3 increased the LogL to 2611.71, a substantial gain. The XFA2 model in part 4 increased the LogL to 2621.58. Finally, fitting US in part 5 resulted in a LogL of 2620.59. This has only one more parameter than the XFA2 model which has 4 more free parameters than the XFA1 model. Consequently, the XFA2 model is the best fit. However, since one specific variance becomes zero, the Akaike report discounts it

```
PART 1: Akaike Information Criterion    -5065.32 (assuming 31 parameters).
PART 2: Akaike Information Criterion    -5111.39 (assuming 31 parameters).
PART 3: Akaike Information Criterion    -5153.42 (assuming 35 parameters).
PART 4: Akaike Information Criterion    -5167.16 (assuming 38 parameters).
PART 5: Akaike Information Criterion    -5164.11 (assuming 40 parameters).
```

In this data the unstructured variance matrix can be fitted but in general, especially with more than five sites, with high genetic correlations and with fewer genotypes, the REML estimate of an unstructured variance matrix, if it can be successfully estimated, may not be a positive definite matrix. The final genetic variance matrix for the US was

```
Covariance/Varianc/Correlation Matrix US us(Site).Genotype
0.5948E-01 0.6094      0.7278      0.5225      0.7740
0.4757E-01 0.1025      0.6042      0.5910      0.6718
0.2111E-01 0.2300E-01 0.1415E-01 0.4820      0.7916
0.4938E-01 0.7331E-01 0.2222E-01 0.1502      0.4434
0.4662E-01 0.5310E-01 0.2325E-01 0.4243E-01 0.6098E-01
```

Recall, the average correlation from the CORUH model was 0.626.

It is convenient at this point to explore the XFA model which is akin to Principal Components analysis. The underlying, latent variables are called factors. The XFA1 model assumes a single factor, a set of genotype effects, usually called genotype scores, that explain the covariance among sites. The XFA2 assumes two genotype factors. The XFA2 model forms the across site variance matrix as  $\Gamma\Gamma' + \Psi$  with estimates from this example of

$$\Psi = \begin{pmatrix} 0.0172 & 0 & 0 & 0 & 0 \\ 0 & 0.0429 & 0 & 0 & 0 \\ 0 & 0 & 0.0041 & 0 & 0 \\ 0 & 0 & 0 & 0.0000 & 0 \\ 0 & 0 & 0 & 0 & 0.0076 \end{pmatrix} \text{ and } \Gamma = \begin{pmatrix} 0.089 & 0.185 \\ 0.056 & 0.237 \\ 0.049 & 0.089 \\ -0.170 & 0.348 \\ 0.134 & 0.188 \end{pmatrix}$$

The elements in the diagonal matrix  $\Psi$  are known as specific variances and represent the variation in genotype effects that is site specific (not associated with the factors). The elements of  $\Gamma$  are the

loadings for the two factors at the five sites. In the XFA2 case to maintain identifiability, ASReml in each iteration will not update the first element of the second factor but will then rotate the loadings to orthogonality. They represent the regression of the genotype effects for each site on the latent factors which have no intrinsic meaning. Plotting them maps the sites according to genetic similarity highlighting that one site (4 at coordinates -0.170, 0.348) is most divergent from the others as shown in Figure 16.8.

The XFA2 genetic variance matrix as presented in the .asr file is

```
Covar/Var/Corr Matrix XFA xfa(Site,2).Genotype
0.5946E-01 0.6268 0.7110 0.5223 0.7763 0.3652 0.7595
0.4885E-01 0.1022 0.6191 0.5901 0.6585 0.1739 0.7416
0.2081E-01 0.2375E-01 0.1441E-01 0.4831 0.7853 0.4112 0.7383
0.4938E-01 0.7313E-01 0.2248E-01 0.1503 0.4442 -0.4387 0.8986
0.4671E-01 0.5195E-01 0.2326E-01 0.4250E-01 0.6090E-01 0.5446 0.7602
0.8905E-01 0.5559E-01 0.4936E-01 -0.1701 0.1344 1.000 0.000
0.1852 0.2371 0.8863E-01 0.3484 0.1876 0.000 1.000
```

The first 5×5 block is the variance matrix, directly comparable to the US matrix displayed above, with derived correlations in the upper right triangle. The first 5 columns of the last 2 rows are the loadings,  $\Gamma'$ , being the covariances of the genotypes effects at the 5 sites with the latent factors. Similarly, the first 5 rows of the last 2 columns are the correlations of the genotypes effects at the 5 sites with the latent factors. The final 2 × 2 identity matrix relates to the two factors in that they are intended to be uncorrelated.

The .sln file also contains genotype effects for the five sites and two factors, that is the genotype scores, as well as genotype effects for the 5 sites. For example

```
xfa(Site,2).Genotype BLA3071.VV5866 0.1007 0.9392E-01
xfa(Site,2).Genotype MTA3071.VV5866 0.1233 0.1675
xfa(Site,2).Genotype PNA3071.VV5866 0.7306E-01 0.5347E-01
xfa(Site,2).Genotype RSA3071.VV5866 -0.3656 0.1640
xfa(Site,2).Genotype WTA3071.VV5866 0.2092 0.8412E-01
xfa(Site,2).Genotype Factor_1.VV5866 1.658 0.5519
xfa(Site,2).Genotype Factor_2.VV5866 -0.2401 0.3483
```

The reported genotype E-BLUPs for each site are calculated as  $\Gamma \mathbf{s} + \tilde{\delta}$  where  $\mathbf{s}$  is the score  $\begin{pmatrix} 1.658 \\ -0.240 \end{pmatrix}$  and  $\tilde{\delta}$  is a lack of fit residual (with variance given by  $\Psi$ ). Note that the residuals  $\tilde{\delta}$  are not explicitly returned by ASReml. Thus

$$\begin{pmatrix} 0.1007 \\ 0.1233 \\ 0.0731 \\ -0.3656 \\ 0.2092 \end{pmatrix} = \begin{pmatrix} 0.089 & 0.185 \\ 0.056 & 0.237 \\ 0.049 & 0.089 \\ -0.170 & 0.348 \\ 0.134 & 0.188 \end{pmatrix} \begin{pmatrix} 1.658 \\ -0.240 \end{pmatrix} + \begin{pmatrix} -0.0024 \\ 0.0874 \\ 0.0132 \\ 0.0002 \\ 0.0322 \end{pmatrix}$$

The factors will be orthogonal if the user has not applied explicit constraints to the loadings, or they can be rotated to be orthogonal. Plotting the genotype scores on the factor axes gives a two-dimensional representation of them. Plotting the loadings gives information on the similarity of sites with respect to genotype ranking. Plotting both together provides a biplot. Note that biplots are only useful when the factors plotted explain a large proportion of the variation. The average genotype ranking is given by predicting genotype effects (BLUPs) at the average loadings (0.0316,

0.2094). Stability of genotype performance is assessed by evaluating them at the extreme sites. The choice of weights to use to produce an index upon which to rank genotypes is beyond our present purpose. The following PREDICT statements simply demonstrate how one might proceed.

```
PREDICT Site Genotype # Predicts each Genotype at each Site.
PREDICT Genotype !AVERAGE Site {1 1 1 0 1}/4
PREDICT Genotype !AVE Site 5*0 0.213 0.0316 0.2094 !ONLY xfa2(Site).Geno
```

The first PREDICT statement gives predicted means for each genotype at each site, even though some genotypes might not have been grown at some sites. While all sites are statistically distinct in this case, one might be interested in the average of the most similar sites (excluding site 4). This is given in the second prediction. Note that these first two predictions incorporate the `Site.Genotype` BLUPs and so include the site specific residuals ( $\tilde{\delta}$ ). The third prediction is based on the genotype scores only and predicts the common genotype effects at the average environment (the average of the loadings). Note the `!ONLY` qualifier; without it, the predictions are not estimable because they include the  $\mu$  term but not `Site` fixed effects.

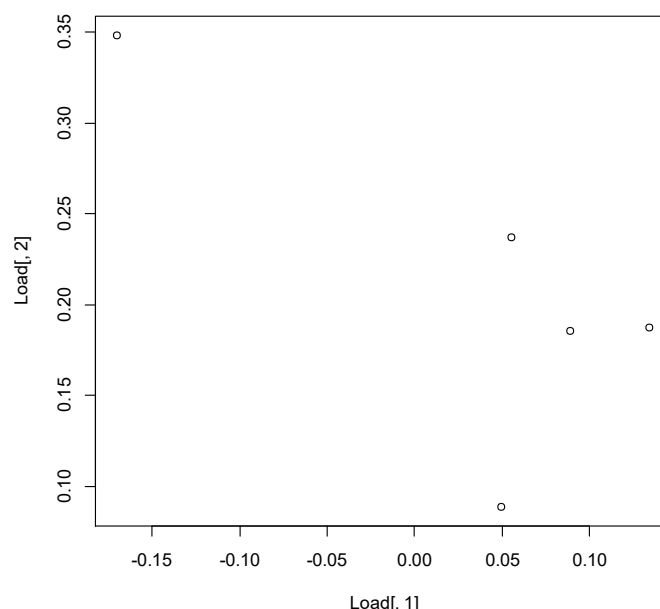


Figure 16.8: Plot of first and second loadings for XFA2 model (part 4)

## 16.8.2 Meta analysis of trial means

When it comes to later stages of selection, it is desirable to include as many trials as possible representing different locations and seasons when comparing genotypes. This will commonly be over 30 experiments after 3 years of evaluation, and could easily be 100-200 trials. Furthermore, few genotypes will be represented at all sites. The combined analysis of trials at the plot level discussed in the previous subsection is therefore not always feasible.

We describe a two-step approach. First, each experiment is analysed separately under a spatial model and predicted genotype means are produced along with a set of weights (see `!TWOSTAGEWEIGHTS` on page 180). Ideally, these should be stored in a database so that they

can be conveniently retrieved later for subsequent analysis. We also store the site mean yield and the residual variance along with details of the spatial model fitted.

For this example, we have extracted 2,184 predicted wheat yields from the original database, with their weights. They represent 273 genotypes conducted in 8 locations. Approximately 0.27% of the cells are missing. The yields range from 4.84 to 121.78 with an average of 61.95. The trial variances range 81.763 to 225.343 with an average of 131.316. Following is the code that fits 5 models to this data.

```
!RENAME !ARG 11 1 2 3 !CONTINUE
Wheat - Two-stage analysis
  location !A
  gen !A
  yield
  weight
pheno_wheat_preds.csv !SKIP 1 !AILOAD 40 !EXTRA 5 !DOPART $1

!PART 11      # Initial Model - Uniform correlation, heterogenous variance
yield !WT = weight ~ mu location !r corh(location).gen
residual idv(units 1 !GF)

!PART 1       # FA of order 1
yield !WT = weight ~ mu location !r xfa1(location).gen
residual idv(units 1 !GF)

!PART 2       # FA of order 2
yield !WT = weight ~ mu location !r xfa2(location).gen
residual idv(units 1 !GF)

!PART 3       # FA of order 3
yield !WT = weight ~ mu location !r xfa3(location).gen
residual idv(units 1 !GF)
PREDICT location gen 2
```

The equal correlation model is fitted first in Part 11. After 15 iterations, it had converged with a LogL of -6074.96 with an average genetic correlation of 0.475. Continuing with XFA1 in part 1, the LogL increases to -6028.40 in 15 iterations. The factor explains 54.1% of the genotype variation. Again, using !CONTINUE to start with the XFA1 values, Part 2 fits an XFA2 model. After 24 iterations, the LogL had reached -5997.81 with 1 specific variances fixed at 0.0. The second factor explains a further 16.4% of the genotype variation. The parameters were still changing slightly but this provides two factors and permits a biplot of genotypes and experiments to be formed.

XFA models with two or more factors are often difficult to fit. The main strategy for fitting these models is getting better starting values which is why these 4 models are fitted in sequence and using !CONTINUE. The !AILOAD  $f$  qualifier may also help. It is set by default when XFA $k$  model parameters are initialized using !CONTINUE from a previous XFA model fit with  $k - 1$  factors. The strategy followed, when the user does not supply explicit constraints and !AILOAD  $f$  is set, has three stages. The first stage is to hold the first  $k - 1$  factor loadings fixed for a few iterations, estimating the loadings for the  $k$ th factor and the specific variances. The second stage is to estimate two factors and the specific variances until iteration  $f$  before updating all factors simultaneously in the third stage. There is also an automatic procedure which shrinks the AI updates for the loadings as a set if as a set they appear relatively large.

Moving from XFA2 to XFA3, the LogL increased 10.62 (-5997.81 to -5987.19) with 8 extra parameters (for a total of  $8 \times 4 = 32$  parameters, but some are constrained). Proceeding to XFA4 or farther is probably ambitious, but still possible, and it should be more difficult if a good portion of the location  $\times$  genotype cells are empty, and when more sites/environments are considered. In addition, for this example, given the low number of locations (8), it might not be justified to go to higher orders.

The summary of the results are

```
XFA1  LogL -6028.40      AIC  12088.79 (16)
XFA2  LogL -5997.81      AIC  12039.62 (22)
XFA3  LogL -5987.19      AIC  12030.37 (28) (lowest AIC)
```

Note that the calculation of AIC is based on ‘free parameters’ and excludes values zero on the specific variances ( $\Psi$ ) and identifiability constraints.

The interpretation of results from a factor analytic model is not easy, but is similar to interpreting results from a principal components analysis. First, users are referred to the section of the .res file headed DISPLAY of variance partitioning for XFA structure in ... which lists the specific variance and loadings as a table with a figure (see page 232). Further investigation must be done outside of ASReml using the variance parameters and the fitted effects.

### Using R to look for patterns in the loadings

The first step is to extract the specific variance and loadings from the .msv file (say wheat3.msv in this example based on the run for XFA3).

```
MSV <- read.table("wheat3.msv", header = FALSE, sep = ",",
                  comment.char = "#", skip = 9)
```

which gives a data frame as

```
5          Variance 1  V  F  1.00000  5  1
6          idv(units);idv(units)_1  G  F  1.00000  6  1
7  xfa3(location).gen;xfa3(location)_1  V  P  16.69751  7  1
:
36 xfa3(location).gen;xfa3(location)_30  L  P  1.0550827  36  1
37 xfa3(location).gen;xfa3(location)_31  L  P -2.8619629  37  1
38 xfa3(location).gen;xfa3(location)_32  L  P -0.8780977  38  1
```

Then, we can extract from this data frame the specific variance and loadings.

```
XFA <- matrix(MSV$V5[3:34], 8, 4)
dimnames(XFA) <- list(paste('S', 1:8, sep=''),
                     c("Psi", "Load_1", "Load_2", "Load_3"))
```

The code from above is reading the column of estimated variance components with specific variances first, then followed but the three loadings, and this is assigned to a matrix of 8 rows (sites) and 4 columns. Hence, giving the following matrix XFA

```
      Psi  Load_1  Load_2  Load_3
S1 16.6975120  7.057940 -0.2638369  2.5059130
S2 55.6495420  5.426707 -3.0906606 -0.4417835
S3  0.5093471  7.614685  1.0571018  2.2801249
```

## 16.8 Multi-Environment Trials

```
S4 0.0000000 8.648089 -4.4077741 -2.8618692
S5 23.5127510 7.529468 -2.8490788 1.1792347
S6 86.9775050 6.874391 4.2330772 1.0550827
S7 18.4901070 6.477795 5.6705408 -2.8619629
S8 71.5190000 6.257515 0.6908808 -0.8780977
```

The use of the R command `pairs(XFA)` will provide the following figure that contains the specific variances in column 1.

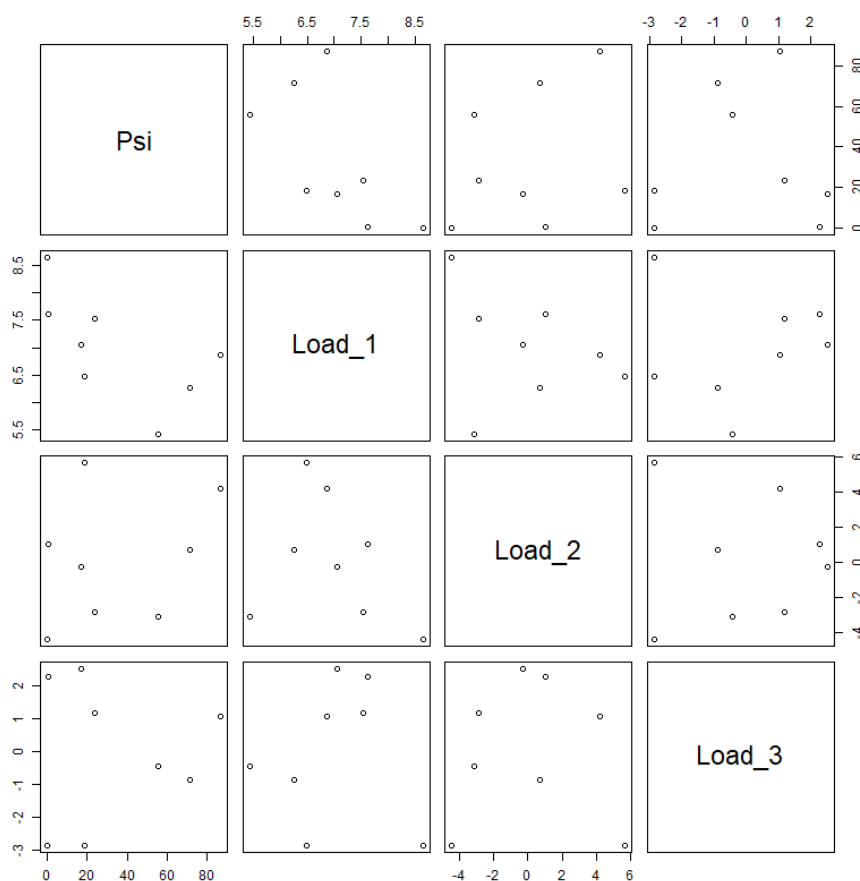


Figure 16.9: Scatterplot of specific variance and loadings for the wheat dataset

A prediction of the variety Freeman at each of the locations is given by

```
PREDICT location gen 2
```

where the number 2 identifies the second level of factor gen.

```
Predicted values of yield
```

```
gen is evaluated at Freeman
```

```
Warning:      1 non-estimable [aliased] cell(s) may be omitted from the table.
              The Overall SED statistic includes non-estimable predictions.
```

| location    | Predicted Value | Standard Error | Ecode |
|-------------|-----------------|----------------|-------|
| Alliance    | 81.8986         | 0.9465         | E     |
| Clay_Center | 45.3673         | 0.8891         | E     |
| Grant_D     | 87.9263         | 1.5916         | E     |
| Lincoln_IM  | 98.4373         | 2.4974         | E     |



```

McCook          92.9721          2.5929 E
North_Platte    81.3979          1.5590 E
Sidney          74.6755          0.9203 E
SED: Overall Standard Error of Difference    2.398

```

## 16.9 The reduced animal model (RAM)

The reduced animal model (RAM) was devised to reduce the computation involved in fitting a large animal model. The underlying principles discussed here are particularly relevant for forestry trials where genetic links between trials are all through the parents. We consider the case where there is at most one record per individual, a large proportion of the individuals are non-parents (they have no progeny) and there is interest in predictions for parents alone.

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 for a tree model for a single trait and a single site might be

```

DBH ~ mu !r nrmv(tree) plot arlv(column).arl(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 along with the `arlv(column).arl(row)` component so that the ‘nugget’ variance is not transferred to the `nrmv(tree)` term since trees are unreplicated.

This analysis requires a pedigree file, say `TreePed.csv`, to define the links. To fit a reduced animal model, we need to fit only the parents in the model but incorporate any inbreeding in the non-parents in a mendelian sampling term. If you process the full tree pedigree in **ASReml** with the `!DIAG` qualifier, the resulting `.aif` file will contain the inbreeding level for every tree in the pedigree, the diagonal of the  $A^{-1}$  matrix and a NonParent/Parent classification for every tree. The file looks like

```

Identity      Inbreeding  Diag_of_Ainverse Parent/NonParent
  4    0.0000      2.5000      Parent
  1    0.0000      3.5000      Parent
    :
 148    0.0000      2.1333     NonParent
 149  0.31250E-01    2.0000     NonParent
 150  0.25000      2.0000     NonParent

```

To create the data files needed for the RAM model, 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 ‘ISparent’, setting it to ‘0’ for the progeny records.

Adding the `!GIV 2` qualifier to the pedigree line will also produce a pedigree file for the parents, say `Parent.ped` to use to provide pedigree connections among the parents. It also creates a `.aif` file for the offspring genotypes containing the `Diag_of_Ainverse` values which could be used to help estimate the genetic variance.

We still need some data manipulation using `Diag_of_Ainverse` and `Parent/NonParent`. This can be done with **ASReml** transformations but is easier if these appear as the first variables in the data file (use R or Excel to reorder the variables). Let’s assume our data file `ramdbh.txt` now has fields: P/NP, AIdiag, tree, mum, dad, row, column, plot, DBH,

## 16.9 The reduced animal model (RAM)

---

parent. 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
IsNProw !A !L NonParent Parent !=1 # Replaces P_NP with binary (0,1)IsNProw
AIdiag !*IsNProw
tree
mum !P !*IsNProw
dad !P !*IsNProw
row *
column *
plot *
DBH
parent !P
WT !=0.125 !+AIdiag !^-1 !*AIdiag !+1 !-IsNProw
Parent.ped
ramdbh.csv
DBH !WT WT ~ mu,
!r nrm(parent !INIT 0.125 !GF) and(mum,0.5) and(dad,0.5),
idv(plot) arlv(column).ar1(row)
residual idv(units)
```

In this model

- `IsNProw !A !L Nonparent Parent !=1`  
codes the nonparent rows with 1, the parent rows as 2 and then recodes them as 1,0 so that `IsNProw` can be used to change some fields associated with parent to zero.
- `!*IsNProw`  
transformations put mum, dad and `AIdiag` information to zero for parents.
- `WT !=0.125 !+AIdiag !^-1 !*AIdiag !+1 !-IsNProw`  
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  $\gamma$  is the variance ratio  $\sigma_g^2/\sigma_e^2$ , 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 all elements zero.
- 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.

## 16.10 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 16.10](#) 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 16.10](#)). 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.

this is for the paired data  
 Y-axis:  $y = \text{sye}$  1.8957 14.8835 X-axis:  $x = \text{sync}$  8.2675 23.5051

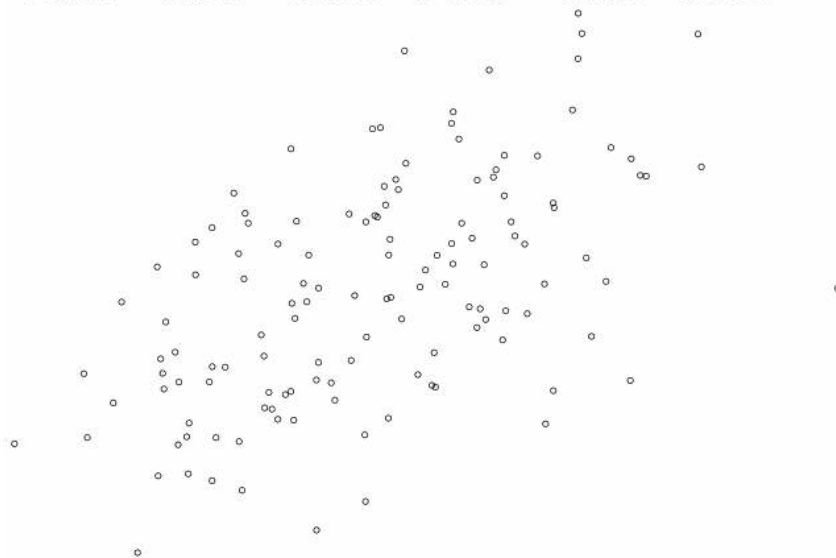


Figure 16.10: Rice bloodworm data: Plot of square root of root weight for treated versus control

### 16.10.1 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 `sqr(rootwt)` for control and treated. The model for each trait is given by

$$\mathbf{y}_j = \mathbf{X}\boldsymbol{\tau}_j + \mathbf{Z}_v\mathbf{u}_{v_j} + \mathbf{Z}_r\mathbf{u}_{r_j} + \mathbf{e}_j \quad (j = c, t) \quad (16.9)$$

where  $\mathbf{y}_j$  is a vector of length  $n = 132$  containing the `sqrroot` values for variate  $j$  ( $j = c$  for control and  $j = t$  for treated),  $\boldsymbol{\tau}_j$  corresponds to a constant term and  $\mathbf{u}_{v_j}$  and  $\mathbf{u}_{r_j}$  correspond to random variety and run effects. The design matrices are the same for both traits. The random effects and error are assumed to be independent Gaussian variables with zero means and variance structures  $\text{var}(\mathbf{u}_{v_j}) = \sigma_{v_j}^2 \mathbf{I}_{44}$ ,  $\text{var}(\mathbf{u}_{r_j}) = \sigma_{r_j}^2 \mathbf{I}_{66}$  and  $\text{var}(\mathbf{e}_j) = \sigma_j^2 \mathbf{I}_{132}$ . The bivariate model can be written as a direct extension of (16.9), namely

$$\mathbf{y} = (\mathbf{I}_2 \otimes \mathbf{X}) \boldsymbol{\tau} + (\mathbf{I}_2 \otimes \mathbf{Z}_v) \mathbf{u}_v + (\mathbf{I}_2 \otimes \mathbf{Z}_r) \mathbf{u}_r + \mathbf{e}^* \quad (16.10)$$

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

There is an equivalence between the effects in this bivariate model and the univariate model of (16.7). The variety effects for each trait ( $\mathbf{u}_v$  in the bivariate model) are partitioned in (16.7) into variety main effects and `tmt.variety` interactions so that  $\mathbf{u}_v = \mathbf{1}_2 \otimes \mathbf{u}_1 + \mathbf{u}_2$ . There is a similar partitioning for the run effects and the errors (see Table 16.10).

In addition to the assumptions in the models for individual traits (16.9) the bivariate analysis involves the assumptions  $\text{cov}(\mathbf{u}_{v_c}) \mathbf{u}'_{v_c} = \sigma_{v_{ct}} \mathbf{I}_{44}$ ,  $\text{cov}(\mathbf{u}_{r_c}) \mathbf{u}'_{r_t} = \sigma_{r_{ct}} \mathbf{I}_{66}$  and  $\text{cov}(\mathbf{e}_c) \mathbf{e}'_t = \sigma_{ct} \mathbf{I}_{132}$ . Thus, random effects and errors are correlated between traits. So, for example, the variance matrix for the variety effects for each trait is given by

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

This unstructured form for `trait.variety` in the bivariate analysis is equivalent to the `variety` main effect plus heterogeneous `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.

```
Paired data - Rice
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

```
:
5 LogL= -344.180      S2=  1.0000      262 df
6 LogL= -343.250      S2=  1.0000      262 df
7 LogL= -343.221      S2=  1.0000      262 df
8 LogL= -343.220      S2=  1.0000      262 df
9 LogL= -343.220      S2=  1.0000      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    2.14373      2.14373      4.44      0 P
Trait      US_C  2  1    0.987412     0.987412     2.59      0 P
Trait      US_V  2  2    2.34754      2.34754      4.62      0 P
us(Trait).id(variety)                     88 effects
Trait      US_V  1  1    3.83960      3.83960      3.47      0 P
Trait      US_C  2  1    2.33401      2.33401      3.01      0 P
Trait      US_V  2  2    1.96182      1.96182      2.69      0 P
us(Trait).id(run)                        132 effects
Trait      US_V  1  1    1.70787      1.70787      2.62      0 P
Trait      US_C  2  1    0.319102     0.319102     0.59      0 P
Trait      US_V  2  2    2.54315      2.54315      3.20      0 P
Covariance/Variance/Correlation Matrix US Residual
2.144      0.4402
```

## 16.10 Paired Case-Control study – Rice

```

0.9874      2.348
Covar/Var/Corr Matrix US us(Trait).id(variety)
  3.839      0.8504
  2.333      1.961
Covariance/Variance/Correlation Matrix US us(Trait).id(run)
  1.708      0.1533
  0.3197     2.544

```

| Source of Variation | Wald F statistics<br>NumDF | F-inc  |
|---------------------|----------------------------|--------|
| 9 Trait             | 2                          | 862.80 |

The resultant **REML** log-likelihood is identical to that of the heterogeneous univariate analysis (column (b) of [Table 16.9](#)). The estimated variance parameters are given in [Table 16.8](#).

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(ye)      14.9532      0.9181 E      223.5977      27.4568
AliCombo     sqrt(ye)      7.9941      0.7993 E      63.9053      12.7795
Bluebelle    sqrt(ye)      13.1033     0.9310 E      171.6963      24.3977
Bluebelle    sqrt(ye)      6.6298      0.8062 E      43.9541      10.6903
C22          sqrt(ye)      16.6678     0.9181 E      277.8168      30.6052
C22          sqrt(ye)      8.9544      0.7993 E      80.1806      14.3146
Chiyohikari  sqrt(ye)      17.3393     0.9301 E      300.6498      32.2534
Chiyohikari  sqrt(ye)      9.6434      0.8057 E      92.9946      15.5402
:
YRK1         sqrt(ye)      15.1860     0.9549 E      230.6135      29.0011
YRK1         sqrt(ye)      8.3357      0.8190 E      69.4833      13.6541
YRK3         sqrt(ye)      13.3059     0.9549 E      177.0472      25.4107
YRK3         sqrt(ye)      8.1134      0.8190 E      65.8278      13.2901
SED: Overall Standard Error of Difference      1.215

```

**Table 16.8:** 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       |

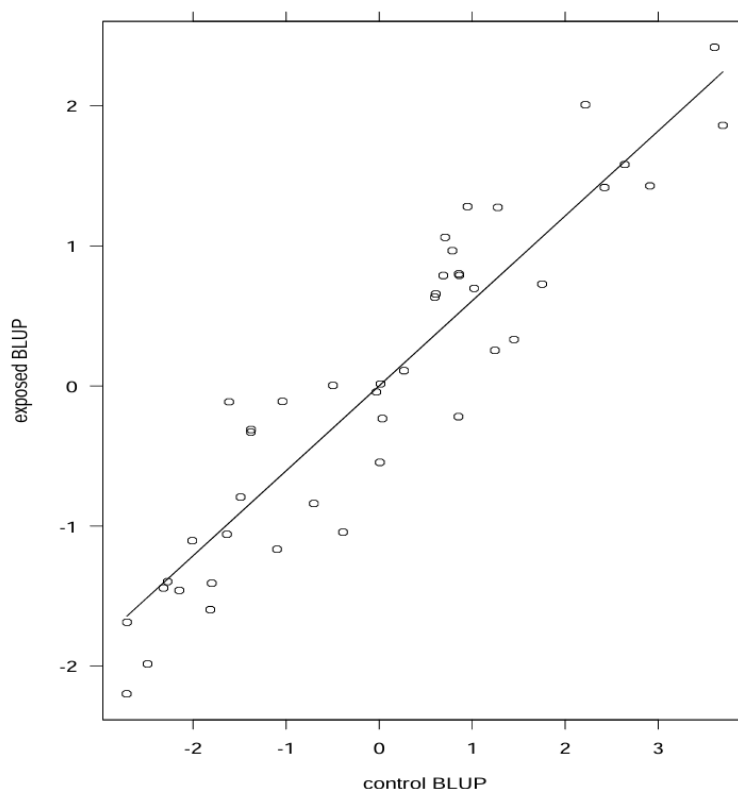


Figure 16.11: BLUPs for treated for each variety plotted against BLUPs for control

### 16.10.2 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  $\text{sqrt}(\text{rootwt})$  but rather two traits, namely  $\text{sqrt}(\text{rootwt})$  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  $\text{sqrt}(\text{rootwt})$  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

$$\mathbf{y} = \mathbf{X}\boldsymbol{\tau} + \mathbf{Z}_1\mathbf{u}_1 + \mathbf{Z}_2\mathbf{u}_2 + \mathbf{Z}_3\mathbf{u}_3 + \mathbf{Z}_4\mathbf{u}_4 + \mathbf{Z}_5\mathbf{u}_5 + \mathbf{e} \quad (16.7)$$

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

The ASReml code for this analysis is

```
Bloodworm data - Dr M. Stevens
id
pair 132
run 66
variety 44 !A
tmt 2 !A
rootwt
sqroot # temp
ricelong.csv !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 !GU).id(variety) idv(run),
idv(pair) diag(tmt !GUU).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 part 1 first. A portion of the output file is

```
:
6 LogL= -345.262      S2=  1.3160          262 df      1.750      0.2825      0.3744
  0.7322      1.329
7 LogL= -345.256      S2=  1.3151          262 df      1.793      0.2549      0.3744
  0.7398      1.329
8 LogL= -345.256      S2=  1.3150          262 df      1.805      0.2469      0.3744
  0.7416      1.329
Final parameter values
  0.7421      1.329
```



## 16.10 Paired Case-Control study – Rice

```

- - - Results from analysis of sqrt(rootwt) - - -
Akaike Information Criterion      702.51 (assuming 6 parameters).
Bayesian Information Criterion    723.92

Approximate stratum variance decomposition
Stratum      Degrees-Freedom      Variance      Component Coefficients
idv(variety)      44.41      26.1091      7.3      3.0      3.7      2.0      1.5      1.0
idv(run)          45.15      7.40753      0.0      3.5      -0.0      2.0      1.7      1.0
idv(variety.tmt)  39.53      2.99905      0.0      0.0      2.7      0.0      0.2      1.0
idv(pair)         41.43      3.26859      0.0      0.0      0.0      2.0      -0.0      1.0
idv(run.tmt)      52.38      5.12180      0.0      0.0      0.0      0.0      2.2      1.0
Residual Variance 39.09      1.31499      0.0      0.0      0.0      0.0      0.0      1.0

Model_Term      IDV_V      Gamma      Sigma      Sigma/SE      % C
idv(variety)      IDV_V      44      1.80824      2.37781      3.01      0 P
idv(run)          IDV_V      66      0.244654      0.321717      0.59      0 P
idv(variety.tmt)  IDV_V      88      0.374379      0.492303      1.78      0 P
idv(pair)         IDV_V      132     0.742082      0.975828      2.51      0 P
idv(run.tmt)      IDV_V      132     1.32916      1.74782      3.65      0 P
units            SCA_V      264     1.00000      1.31499      4.42      0 P

Wald F statistics
Source of Variation      NumDF      DenDF      F-inc      P-inc
9 mu                      1          53.6      1485.65      <.001
5 tmt                     1          60.4      469.33      <.001

```

Table 16.9: Estimated variance components from univariate analyses of bloodworm data. (a) Model with homogeneous variance for all terms and, (b) Model with heterogeneous variance for interactions involving tmt

| source              | (a)      | (b)     |         |
|---------------------|----------|---------|---------|
|                     |          | control | treated |
| variety             | 2.378    | 2.334   |         |
| variety.trt         | 0.492    | 1.506   | -0.372  |
| run                 | 0.322    | 0.319   |         |
| run.trt             | 1.748    | 1.389   | 2.224   |
| variety.run (pair)  | 0.976    | 0.987   |         |
| tmt.pair            | 1.315    | 1.156   | 1.360   |
| REML log-likelihood | -345.256 | -343.22 |         |

The estimated variance components from this analysis are given in column (a) of Table 16.9. 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  $\text{var}(\mathbf{u}_2) = \sigma_2^2 \mathbf{I}_{88}$  we assume

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

where  $\sigma_{2c}^2$  and  $\sigma_{2t}^2$  are the `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= -343.607      S2=  1.1794      262 df  :   1 components restrained
6 LogL= -343.362      S2=  1.1567      262 df  :   1 components restrained
7 LogL= -343.253      S2=  1.1622      262 df
8 LogL= -343.221      S2=  1.1543      262 df
9 LogL= -343.220      S2=  1.1565      262 df
10 LogL= -343.220     S2=  1.1563      262 df

- - - Results from analysis of sqrt(rootwt) - - -
Akaike Information Criterion      704.44 (assuming 9 parameters).
Bayesian Information Criterion    736.56

Model_Term                      IDV_V  44  Gamma      Sigma      Sigma/SE  % C
idv(variety)                     IDV_V  66  2.01858    2.33408    3.01    0 P
idv(run)                         IDV_V 132  0.275912   0.319036   0.59    0 P
idv(pair)                       IDV_V 264  0.853936   0.987403   2.59    0 P
idv(uni(tmt,2))                 IDV_V 264  0.176264   0.203813   0.32    0 P
units                           SCA_V 264  1.00000    1.15630    2.77    0 P
diag(tmt).id(variety)           88 effects
tmt                             DIAG_V  1  1.30205    1.50556    2.26    0 U
tmt                             DIAG_V  2 -0.321877 -0.372186 -0.82    0 U
diag(tmt).id(run)              132 effects
tmt                             DIAG_V  1  1.20105    1.38877    2.18    0 P
tmt                             DIAG_V  2  1.92338    2.22399    3.07    0 P

Wald F statistics
Source of Variation      NumDF  DenDF  F_inc  P_inc
9 mu                      1      56.4  1276.72 <.001
5 tmt                     1      60.6   448.81 <.001

```

The estimated variance components from this analysis are given in column (b) of [Table 16.9](#). 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

$$\text{var}(\mathbf{1}_2 \otimes \mathbf{u}_1 + \mathbf{u}_2) = \begin{bmatrix} \sigma_1^2 + \sigma_{2c}^2 & \sigma_1^2 \\ \sigma_1^2 & \sigma_1^2 + \sigma_{2t}^2 \end{bmatrix} \otimes \mathbf{I}_{44} \quad (16.8)$$

Using the estimates from Table 16.9, this structure is estimated as

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

Thus, the variance of the variety effects in the control group (also known as the genetic variance for this group) is 3.84. The genetic variance for the treated group is much lower (1.96). The genetic correlation is  $2.33 / \sqrt{3.84 * 1.96} = 0.85$  which is strong, supporting earlier indications of the dependence between the treated and control root area (Figure 16.10).

Table 16.10: Equivalence of random effects in bivariate and univariate analyses

| effects       | bivariate (model 16.10) | univariate (model 16.7)                            |
|---------------|-------------------------|--|
| trait.variety | $\mathbf{u}_v$          | $\mathbf{1}_2 \otimes \mathbf{u}_1 + \mathbf{u}_2$ |
| trait.run     | $\mathbf{u}_r$          | $\mathbf{1}_2 \otimes \mathbf{u}_3 + \mathbf{u}_4$ |
| trait.pair    | $\mathbf{e}^*$          | $\mathbf{1}_2 \otimes \mathbf{u}_5 + \mathbf{e}$   |

### 16.10.3 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

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

where the slope  $\beta = \sigma_{v_{ct}} / \sigma_{v_c}^2$ . Tolerance can be defined in terms of the deviations from regression,  $\epsilon$ . Varieties with large positive deviations have greatest tolerance to bloodworms. Note that this is similar to the researcher's original intentions except that the regression has been conducted at the genotypic rather than the phenotypic level. In Figure 16.11 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.

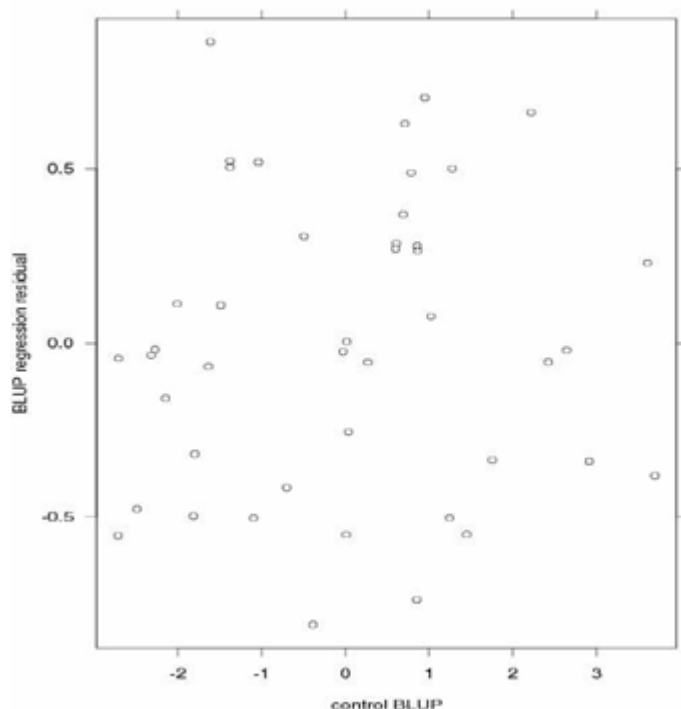


Figure 16.12: Estimated deviations from regression of treated on control for each variety plotted against estimate for control

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

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

Whereas

$$\begin{aligned} \text{cov}(\delta) \mathbf{u}'_{v_c} &= \text{cov}(\mathbf{u}_{v_c} - \mathbf{u}_{v_t}) \mathbf{u}'_{v_c} \\ &= (\sigma_{v_c}^2 - \sigma_{v_{ct}}) \mathbf{I}_{44} \end{aligned}$$

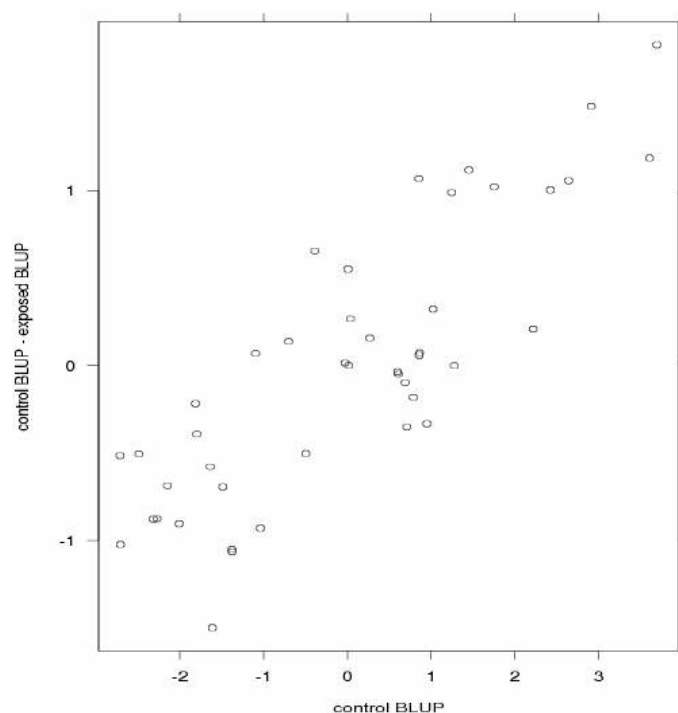


Figure 16.13: Estimated difference between control and treated for each variety plotted against estimate for control

The independence of  $\epsilon$  and  $\mathbf{u}_{v_c}$  and dependence between  $\delta$  and  $\mathbf{u}_{v_c}$  is clearly illustrated in Figure 16.12 and Figure 16.13. 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.

## 16.11 Balanced longitudinal data – 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 re-analysed by Pinheiro and Bates (2000, p338). The data are displayed in Figure 16.14 and are the trunk circumferences (in millimeters) 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

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

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

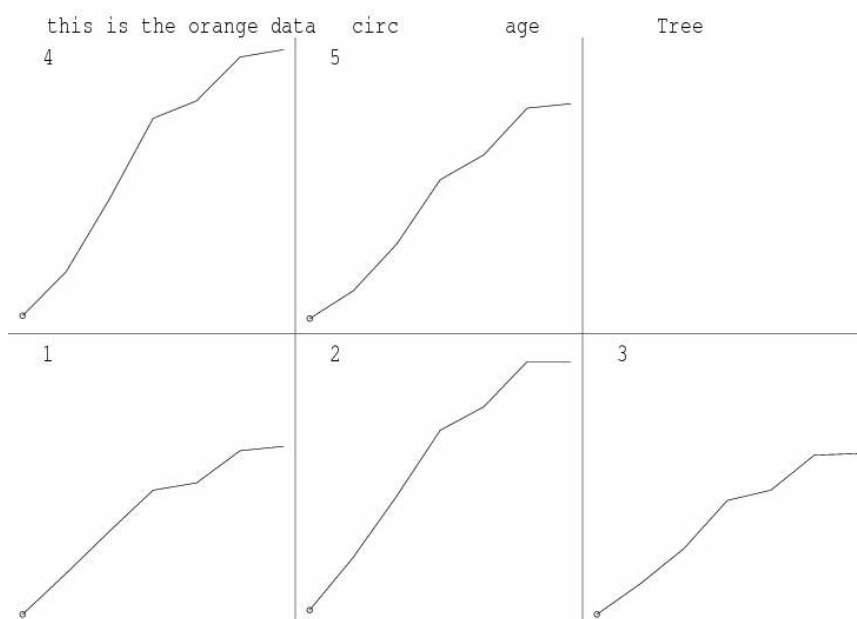


Figure 16.14: Trellis plot of trunk circumference for each tree

The datafile consists of 5 columns, namely, `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 (part 11). 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,  $\delta$  whose elements  $\delta_i, i = 2, \dots, 6$  are the second differentials of the cubic spline at the knot points. The second differentials of a natural cubic spline are zero at the first and last knot points (Green and Silverman, 1994). The **ASReml** job is

```
Orange Data
seq # Record number
Tree 5
age # 118 484 664 1004 1231 1372 1582
circ
season !L Spring Autumn
orange.asd !SKIP 1 !MAXIT 15 !WMF !DOPART $1

!PART 11 # Only tree 1
!FILTER 2 !SELECT 1
!SPLINE spl(age,7) 118 484 664 1004 1231 1372 1582
!PVAL age 150 200:1500
!X age !Y circ !JOIN
circ ~ mu age !r spl(age,7)
residual idv(units)
PREDICT age

!PART 1 2 3 4 5 6
!SPLINE spl(age,7) 118 484 664 1004 1231 1372 1582
!PVAL age 150 200:1500

!PART 1 # Model 1
!X age !Y circ !JOIN
circ ~ mu age !r Tree 4.6 Tree.age 0.000094 spl(age,7) 0.1,
spl(age,7).Tree 2.3
PREDICT age Tree !PLOT

!PART 2 # Model 2
circ ~ mu age !r Tree 4.6 Tree.age 0.000094 spl(age,7) 0.1,
spl(age,7).Tree 2.3 fac(age) 13.9
PREDICT age Tree !PLOT !IGNORE fac(age)

!PART 3 # Model 3
circ ~ mu age season !r Tree 4.6 Tree.age 0.000094 spl(age,7) 0.1,
spl(age,7).Tree 2.3 fac(age) 13.9
PREDICT age Tree !PLOT !IGNORE fac(age)

!PART 4 # Model 4
circ ~ mu age season !r Tree 4.6 Tree.age 0.000094 spl(age,7) 0.1
PREDICT age Tree !PLOT

!PART 5 # Model 5
circ ~ mu age season !r Tree 4.6 Tree.age 0.000094 spl(age,7).Tree 2.3
PREDICT age Tree !PLOT

!PART 6 # Model 6
circ ~ mu age season,
!r str(Tree 4.6 Tree.age us(2 !INIT 4.6 0.00001 0.000094).id(Tree)),
spl(age,7) 0.1 spl(age,7).Tree 2.3
PREDICT age Tree !PLOT
```

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

## 16.11 Balanced longitudinal data – Oranges

! 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    0.1000
2 LogL= -20.9013      S2=  49.152          5 df    0.9102E-01
3 LogL= -20.8998      S2=  50.524          5 df    0.7536E-01
Final parameter values                                0.7937E-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
spl (age,7)      1.50      101.552      12.7      1.0
Residual Variance      3.50      50.5242      0.0      1.0

Model_Term      IDV_V      Gamma      Sigma      Sigma/SE      % C
spl (age,7)      IDV_V      5      0.793732E-01      4.01027      0.41      2 P
units      SCA_V      7      1.00000      50.5242      1.32      0 P

Wald F statistics
Source of Variation      NumDF      DenDF      F-inc      P-inc
7 mu      1      3.5      1373.62      <.001
3 age      1      3.5      216.16      <.001
Note: The DenDF values are calculated ignoring fixed/boundary/singular
variance parameters using algebraic derivatives.

Solution      Standard Error      T-value      T-prev
3 age
1      0.814772E-01      0.554179E-02      14.70
7 mu
1      24.4378      5.77348      4.23
6 spl (age,7)      5 effects fitted

```

The REML estimate of the smoothing constant indicates that there is some nonlinearity. The fitted cubic smoothing spline is presented in Figure 16.15. The fitted values were obtained from the .pvs file. The four points below the line were the spring measurements.

We now consider the analysis of the full dataset. Following Verbyla *et al.* (1999) we consider the analysis of variance decomposition (see Figure 16.13) 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.



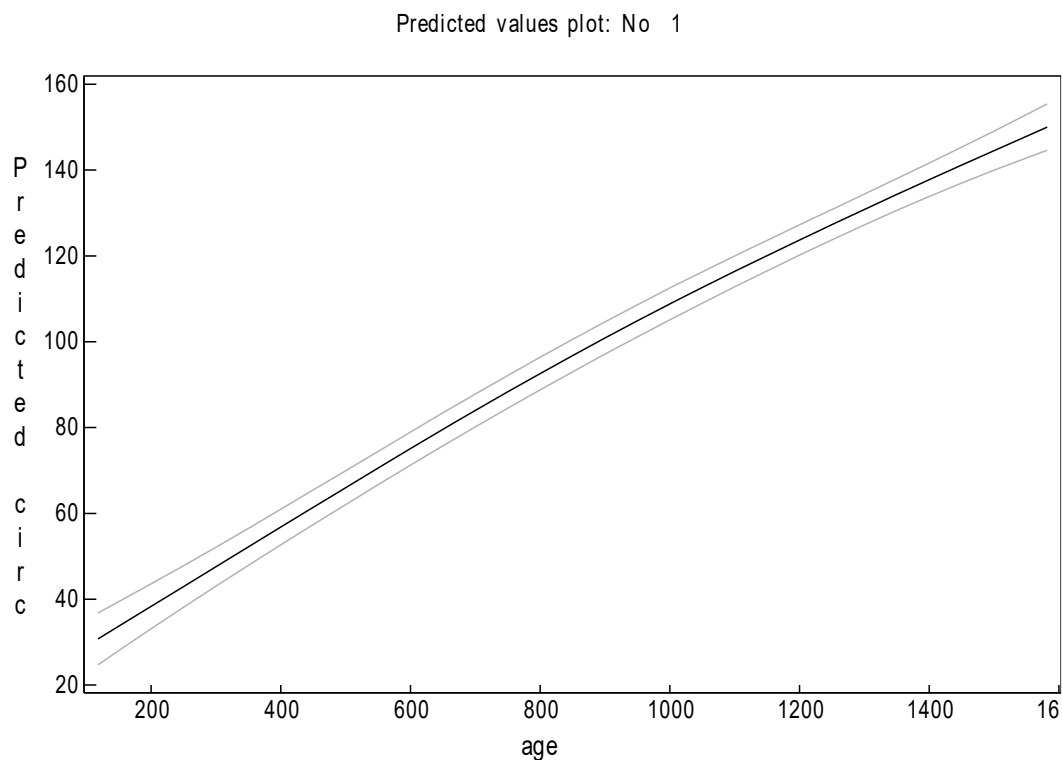


Figure 16.15: Fitted cubic smoothing spline for Tree 1

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 |               |      |          |
|          | age.Tree      | RC   | 5        |
|          | spl (age, 7)  | R    | 25       |
| error    |               | R    |          |

The intercept and slope for each tree are included as random coefficients (denoted by RC in Table 16.11). Thus, if  $\mathbf{U}^{5 \times 2}$  is the matrix of intercepts (column 1) and slopes (column 2) for each tree, then we assume that

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

where  $\mathbf{\Sigma}$  is a  $2 \times 2$  symmetric positive definite matrix. Non smooth variation can be modelled at the overall mean (across trees) level and this is achieved in **ASReml** by inclusion of `fac (age)` as a random term.

Table 16.12: Sequence of models fitted to the Orange data

| model terms         | model  |        |        |        |        |        |
|---------------------|--------|--------|--------|--------|--------|--------|
|                     | 1      | 2      | 3      | 4      | 5      | 6      |
| Tree                | yes    | yes    | yes    | yes    | yes    | yes    |
| age.Tree            | yes    | yes    | yes    | yes    | yes    | yes    |
| (covariance)        | no     | no     | no     | no     | no     | yes    |
| spl (age, 7)        | yes    | yes    | yes    | yes    | no     | yes    |
| Tree.spl (age, 7)   | yes    | yes    | yes    | no     | yes    | yes    |
| fac (age)           | no     | yes    | yes    | no     | no     | no     |
| season              | no     | no     | yes    | yes    | yes    | yes    |
| REML log-likelihood | -97.78 | -94.07 | -87.95 | -91.22 | -90.18 | -87.43 |

An extract of the **ASReml** input file (part 6) is

```
circ ~ mu age season,
!r str(Tree 4.6 Tree.age us(2 !INIT 4.6 0.00001 0.000094).id(Tree)),
spl(age,7) 0.1 spl(age,7).Tree 2.3
```

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,  $\Sigma$ , 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.

```
:
5 LogL= -98.5428      S2= 11.871          33 df      2.590      0.5235E-04  31.31
0.3103
6 LogL= -97.9644      S2=  8.8216          33 df      3.446      0.6999E-04  56.94
0.5570
7 LogL= -97.8142      S2=  7.4477          33 df      4.038      0.8085E-04  79.77
0.7857
8 LogL= -97.7856      S2=  6.8276          33 df      4.409      0.8733E-04  92.21
0.9467
9 LogL= -97.7801      S2=  6.5533          33 df      4.614      0.9089E-04  97.08
1.039
10 LogL= -97.7790      S2=  6.4320          33 df      4.715      0.9264E-04  99.08
1.084
11 LogL= -97.7788      S2=  6.3792          33 df      4.761      0.9344E-04  99.96
1.104
Final parameter values          4.781      0.9379E-04  100.3
1.113
```

- - - Results from analysis of circ - - -

## 16.11 Balanced longitudinal data – Oranges

Akaike Information Criterion 205.56 (assuming 5 parameters).  
 Bayesian Information Criterion 213.04

Approximate stratum variance decomposition

| Stratum           | Degrees-Freedom | Variance | Component | Coefficients |     |
|-------------------|-----------------|----------|-----------|--------------|-----|
| Tree              | 4.09            | 50.0323  | 1.4       | -0.0         | 1.0 |
| spl(age, 7)       | 5.10            | 236.691  | 0.0       | 0.4          | 1.0 |
| spl(age, 7).Tree  | 13.74           | 17.2933  | 0.0       | 0.0          | 1.0 |
| Residual Variance | 6.07            | 6.37917  | 0.0       | 0.0          | 1.0 |

| Model_Term       |       | Gamma | Sigma        | Sigma/SE     | % C  |
|------------------|-------|-------|--------------|--------------|------|
| Tree             | IDV_V | 5     | 4.78112      | 30.4996      | 1.24 |
| Tree.age         | IDV_V | 5     | 0.937853E-04 | 0.598273E-03 | 1.41 |
| spl(age, 7)      | IDV_V | 5     | 100.338      | 640.075      | 1.55 |
| spl(age, 7).Tree | IDV_V | 25    | 1.11324      | 7.10157      | 1.45 |
| Residual         | SCA_V | 35    | 1.00000      | 6.37917      | 1.74 |

| Source of Variation | Wald F statistics |       |
|---------------------|-------------------|-------|
|                     | NumDF             | DenDF |
| 7 mu                | 1                 | 4.0   |
| 3 age               | 1                 | 4.0   |

F\_inc P\_inc  
 47.06 0.002  
 95.01 <.001

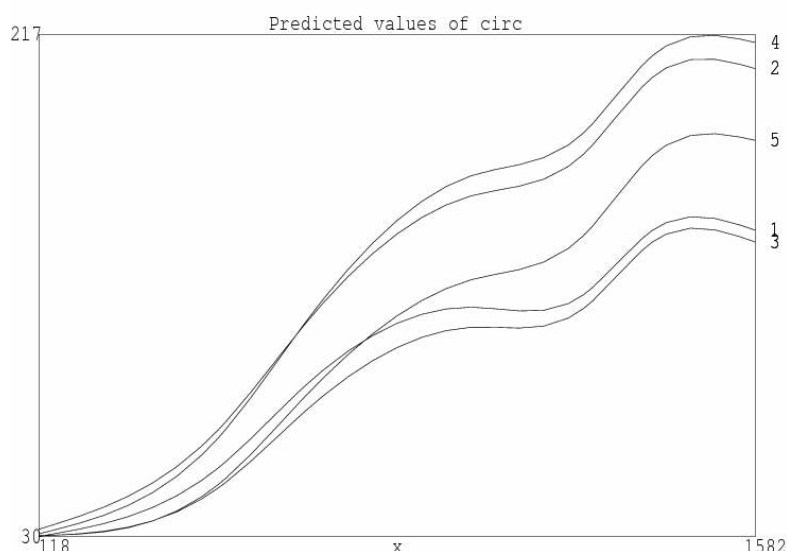


Figure 16.16: Plot of fitted cubic smoothing spline for model 1

A quick look suggests this is fine until we look at the predicted curves in Figure 16.16. 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 boundary. 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

```

:
5 LogL= -87.4407      S2=  5.6838      32 df
6 LogL= -87.4292      S2=  5.5982      32 df
7 LogL= -87.4291      S2=  5.6234      32 df

- - - Results from analysis of circ - - -
Akaike Information Criterion      186.86 (assuming 6 parameters).
Bayesian Information Criterion     195.65
Coefficient of Determination: NDF 2.00 DenDF 4.10 Fall 100.72 CD 98.01

Model_Term              Gamma      Sigma      Sigma/SE      % C
spl(age,7)              IDV_V      5      2.17323      12.2210      1.09      0 P
spl(age,7).Tree          IDV_V      25     1.38721      7.80085      1.47      0 P
Residual                 SCA_V      35     1.00000      5.62340      1.72      0 P
us(2).id(Tree)           10 effects
2                        US_V      1  1      5.62531      31.6334      1.26      0 P
2                        US_C      2  1     -0.124266E-01 -0.698798E-01 -0.85      0 P
2                        US_V      2  2      0.108431E-03  0.609752E-03  1.40      0 P
Covariance/Variance/Correlation Matrix US Tree
 31.70      -0.5031
-0.7002E-01  0.6110E-03

Wald F statistics
Source of Variation      NumDF      DenDF      F_inc      P_inc
7 mu                      1          2.4      169.87      0.003
3 age                     1          2.4      92.78      0.006
5 season                   1          8.8     108.66      <.001

```

Figure 16.17 presents the predicted growth over time for individual trees and a marginal prediction for trees with approximate confidence intervals ( $2 \pm \times$ ) 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 16.18 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.

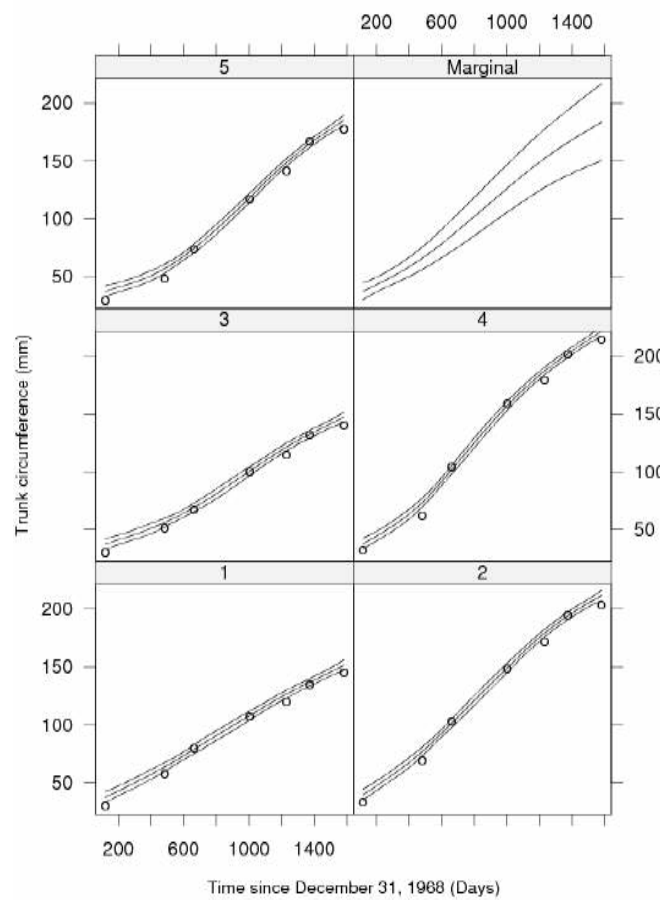


Figure 16.17: Trellis plot of trunk circumference for each tree at sample dates (adjusted for *season* effects), with fitted profiles across time and confidence intervals

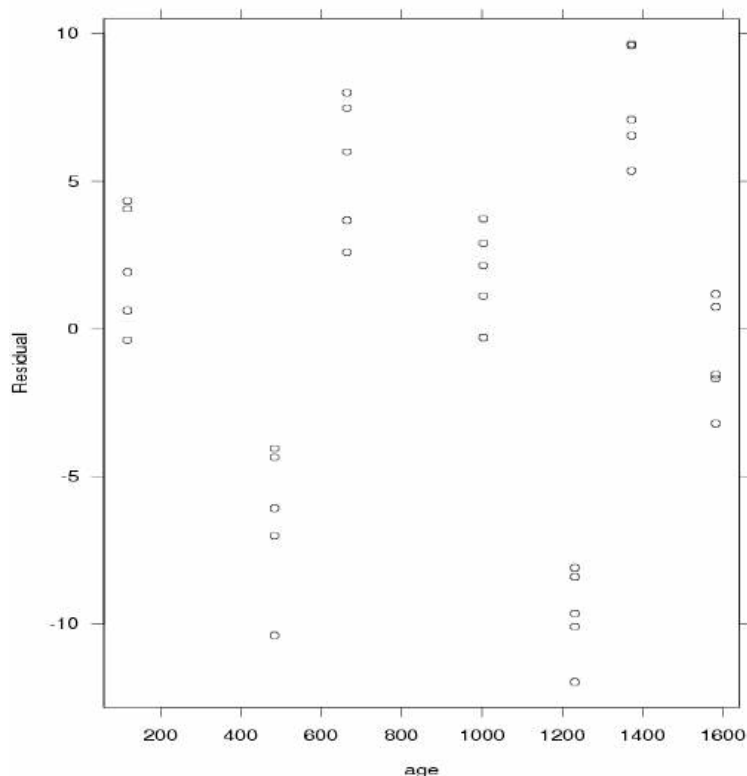


Figure 16.18: Plot of the residuals from the nonlinear model of Pinheiro and Bates

## 16.12 Generalized Linear (Mixed) Models – Sheep

ASReml uses an approximate likelihood technique called penalized quasi-likelihood (PQL) (see Section 6.8 to analyse data sampled from one of the common members of the exponential family. In this section we present a few examples to demonstrate the coding.

### Binomial analysis of Footrot score

Mohammad Alwan (personal communication) for his Master's thesis at Massey University scored the feet of 2,513 lambs born in 1980 and 1981. The lambs were from five mating groups: 7 Perendale rams over Perendale ewes in 1980, 6 Booroola by Romney rams over Perendale ewes in 1980, 3 Booroola rams over Romney ewes in 1980, 6 Perendale rams over Perendale ewes in 1981, and 12 Booroola by Romney rams (from group 3) over Perendale ewes in 1981. This data was previously analysed by Gilmour (1984) and Gilmour et al. (1987).

The data file `lamb.dat` contains grouped data for the 68 combinations of `SEX` and `SIRE` for two foot shape classes: `FS1`, all four feet are normal, `FS2`, one foot is deformed; and two indicator

## 16.12 Generalized Linear (Mixed) Models – Sheep

variables for the presence of disease conditions: Scald and Rot. No scald or rot was present in group 4 lambs and these responses have been set to missing. The data file `alwan.asd` contains the long version of the same data (2,513 records). The genetic relationships among sires are ignored in this analysis although it would just require a sire relationship matrix to include them.

### Normal analysis

Our first analysis is of the incidence of foot rot on the Normal scale. The code is:

```
Mohammad Alwan Lamb Data Foot Scores (long)
```

```
CYR 2
GRP 5
SEX
SIRE !A
TOT
L5
L4
LS
LR
```

```
alwan.asd !SKIP 1 !DDF -1 !SLN !DOPART $1
```

```
LR ~ mu SEX GRP !r SIRE
residual idv(units)
PREDICT GRP 1 2 3 5
```

```
VPREDICT !DEFINE
F Total units + SIRE
F Genetic SIRE*4
H Heritability Genetic Total
```

The associated results for this analysis are

Univariate analysis of LR

Summary of 1960 records retained of 2513 read

| Model term | Size    | #miss | #zero | MinNon0 | Mean       | MaxNon0 | StndDevn |
|------------|---------|-------|-------|---------|------------|---------|----------|
| 1 CYR      | 2       | 0     | 0     | 1       | 1.4179     | 2       |          |
| 2 GRP      | 5       | 0     | 0     | 1       | 2.8679     | 5       |          |
| 3 SEX      |         | 0     | 962   | 1.000   | 0.5092     | 1.000   | 0.5000   |
| 4 SIRE     | 34      | 0     | 0     | 1       | 15.5893    | 34      |          |
| 5 TOT      |         | 0     | 0     | 1.000   | 1.000      | 1.000   | 0.000    |
| 6 L5       |         | 0     | 646   | 1.000   | 0.6704     | 1.000   | 0.4702   |
| 7 L4       |         | 0     | 1392  | 1.000   | 0.2898     | 1.000   | 0.4538   |
| 8 LS       |         | 0     | 1788  | 1.000   | 0.8776E-01 | 1.000   | 0.2830   |
| 9 LR       | Variate | 0     | 1893  | 1.000   | 0.3418E-01 | 1.000   | 0.1817   |
| 10 mu      |         | 1     |       |         |            |         |          |

Forming 41 equations: 7 dense.  
Initial updates will be shrunk by factor 0.316

Note: 2 singularities detected in design matrix.

|         |         |     |             |         |   |                         |
|---------|---------|-----|-------------|---------|---|-------------------------|
| 1 LogL= | 2336.33 | S2= | 0.32368E-01 | 1955 df | : | 1 components restrained |
| 2 LogL= | 2344.75 | S2= | 0.32702E-01 | 1955 df |   | 0.9487E-02              |
| 3 LogL= | 2344.75 | S2= | 0.32707E-01 | 1955 df |   | 0.9168E-02              |
| 4 LogL= | 2344.75 | S2= | 0.32708E-01 | 1955 df |   | 0.9136E-02              |

Final parameter values 0.9132E-02

- - - Results from analysis of LR - - -

Akaike Information Criterion -4685.51 (assuming 2 parameters).

Bayesian Information Criterion -4674.35

| Model_Term | Gamma | Sigma | Sigma/SE | % C |
|------------|-------|-------|----------|-----|
|------------|-------|-------|----------|-----|

## 16.12 Generalized Linear (Mixed) Models – Sheep

```

SIRE          IDV_V   34  0.913205E-02  0.298690E-03  1.27  0 P
units         SCA_V 1960  1.00000      0.327079E-01 31.06  0 P

                                Wald F statistics
      Source of Variation      NumDF      F-inc
10 mu                          1          42.94
3 SEX                          1           0.02
2 GRP                          3           2.05

      Solution      Standard Error      T-value      T-prev
2 GRP
      2  0.235117E-01  0.144864E-01      1.62
      3  0.503927E-01  0.219555E-01      2.30      1.23
      5  0.198901E-01  0.133989E-01      1.48
3 SEX
      1  0.722826E-03  0.820139E-02      0.09
10 mu
      1  0.164059E-01  0.111038E-01      1.48
4 SIRE
                                34 effects fitted (      6 are zero)

SLOPES FOR LOG(ABS(RES)) on LOG(PV) for Section 11
1.13
67 possible outliers in Section 11: see .res file
Finished: 22 Sep 2025 15:39:42.048 LogL Converged

```

Two things stand out in this analysis. From a genetic perspective, the heritability estimate is  $0.0362 = 4 \times 0.000299 / (0.000299 + 0.032707)$  as calculated by the VPREDICT statements. Secondly, there is little evidence of significant difference between sex classes.

The predicted values are

```

PxP 1980      BRxP 1980      BxR 1980      BRxP 1981
0.0168 ± 0.0104  0.0403 ± 0.0101  0.0672 ± 0.0194  0.0367 ± 0.0085

```

Note that the above analyses is obtained using the long version of the data (alwan.asd), the same results can be obtained using the short version of the data (lamb.asd), but in this case additional arguments of !DF and !YSS are required. The code is

```

Mohammad Alwan Lamb Data Foot Scores (short)
CYR 2
GRP 5
SEX
SIRE !A
Total
L5
L4
Scald # LS
Rot # LR
pRot !=Rot !/Total # Convert count to a proportion
lamb.dat !SKIP 1 !EXTRA 5 !DOPART $1

!PART 1
!DF 1904 !YSS 62.54249 !DDF -1
pRot !TOTAL = Total ~ mu SEX GRP !r SIRE 0.16783
residual idv(units)

PREDICT GRP 1 2 3 5
VPREDICT !DEFINE
F GenVar SIRE*4

```



## 16.12 Generalized Linear (Mixed) Models – Sheep

```
F PhenVar SIRE Residual
H heritability GenVar PhenVar
```

### **Binomial analysis**

Binomial analysis typically assumes the variance ( $p(1-p)$ ) can be predicted from the mean ( $p$ ) and is performed on logistic (or probit) scale to prevent the predicted mean going outside the (0, 1) range. Since the variance will be predicted from the mean, we can use the summarized data. The code, using the heading as above with the short data for this Part 2 is

```
!PART 2
pRot !BIN !TOTAL = Total ~ mu SEX GRP !r SIRE
residual id(units)
PREDICT SEX 0 1 GRP 1 2 3 5

VPREDICT !DEFINE
F GenVar SIRE*4
F PhenVar SIRE Residual*3.28987
H heritability GenVar PhenVar
```

A portion of the respective results are

```
Univariate analysis of pRot
Summary of 68 records retained of 68 read

Model term          Size #miss #zero  MinNon0    Mean    MaxNon0  StndDevn
1 CYR                2      0      0      1    1.6176      2
2 GRP                5      0      0      1    3.2941      5
3 SEX                0     34     1.000    0.5000    1.000    0.5037
4 SIRE              34      0      0      1   17.5000    34
5 Total              Weight  0      0   16.00    36.96    67.00   13.10
6 L5                 0      0     6.000    24.06    50.00   10.41
7 L4                 0      0     3.000    10.75    30.00    5.650
8 Scald              0     25     1.000    2.529    16.00    3.348
9 Rot                0     31     1.000    0.9853    4.000    1.139
10 pRot              Variate 0     31 0.1754E-01 0.2970E-01 0.1818  0.3739E-01
11 mu                1
Forming              41 equations:    7 dense.
Initial updates will be shrunk by factor    0.010
* This job uses all of the 4 processor threads. *
Note: Algebraic Denominator DF calculation is not available
      Numerical derivatives will be used.

Distribution and link: Binomial; Logit  Mu=P=1/(1+exp(-XB))
                        V=Mu(1-Mu)/N
                        using weights from variable:    5

Note: The LogL value is unsuitable for comparing GLM models.
Note: 1 singularities detected in design matrix.
1 LogL= -34.0757      S2=  1.0000      62 df  Dev/DF=  0.9641
2 LogL= -47.6679      S2=  1.0000      62 df  Dev/DF=  0.8095
3 LogL= -58.3592      S2=  1.0000      62 df  Dev/DF=  0.7383
4 LogL= -65.1027      S2=  1.0000      62 df  Dev/DF=  0.7031
5 LogL= -65.1553      S2=  1.0000      62 df  Dev/DF=  0.6815
6 LogL= -65.3255      S2=  1.0000      62 df  Dev/DF=  0.6447
7 LogL= -65.3998      S2=  1.0000      62 df  Dev/DF=  0.6383
8 LogL= -65.4088      S2=  1.0000      62 df  Dev/DF=  0.6372
9 LogL= -65.4098      S2=  1.0000      62 df  Dev/DF=  0.6371
10 LogL= -65.4100     S2=  1.0000      62 df  Dev/DF=  0.6371
11 LogL= -65.4100     S2=  1.0000      62 df  Dev/DF=  0.6371
12 LogL= -65.4100     S2=  1.0000      62 df  Dev/DF=  0.6371
13 LogL= -65.4100     S2=  1.0000      62 df  Dev/DF=  0.6371
Final parameter values                                0.2551
```

## 16.12 Generalized Linear (Mixed) Models – Sheep

```

Deviance from GLM fit          62          39.50
Variance heterogeneity factor [Deviance/DF]      0.64

- - - Results from analysis of pRot - - -
Note: While convergence of the LogL value indicates that the model
      has stabilized, its value CANNOT be used to formally test
      differences between Generalized Linear (Mixed) Models.
Akaike Information Criterion      132.82 (assuming 1 parameters).
Bayesian Information Criterion     134.95
Coefficient of Determination: NDF 5.00 DenDF 48.24 Fall 1.54 CD 13.73

Approximate stratum variance decomposition
Stratum      Degrees-Freedom  Variance      Component Coefficients
SIRE          2.99      0.255103          1.0

Model_Term          IDV_V  34  0.255103      0.255103      Sigma/SE      % C
SIRE                  Weight_V  68  1.00000      1.00000      0.00      0 F

Wald F statistics
Source of Variation      NumDF      DenDF      F_inc      P_inc
11 mu                    1        20.3      427.20      <.001
3 SEX                    1        62.0        0.03      0.869
2 GRP                    4        33.5        1.89      0.135
Note: The DenDF values are calculated ignoring fixed/boundary/singular
      variance parameters using numerical derivatives.

Warning: These Wald F statistics are based on the working variable
         and are not equivalent to an Analysis of Deviance.
         Standard errors are scaled by the residual variance of the
         working variable, not the residual deviance.

```

The effects in this analysis are on a logistic scale with a variance of  $3.28987 = \pi^2/3$  and so the heritability on the underlying (logistic) scale is  $0.289 = 4 \times 0.2551/(3.28987+0.2551)$ .

Repeating the analysis on the Probit scale by inserting !PROBIT after !BIN in the model line produces a SIRE component of 0.0506 on the Probit scale which has an underlying variance of 1.0. The heritability estimate is then 0.193. Given the low incidence of the disease (0.034), and low heritability in the Normal scale (0.036), the heritability on the probit scale is expected to be around  $0.206 = 0.0360/(z^2/pq)$  where  $z = 0.0758$  is the ordinate of the Normal(0,1) distribution corresponding to  $p = 1 - q = 0.034$ .

The preceding Wald F Statistics pertain to the working variable created as part of the PQL analysis. The predicted means shown below are not that different from those obtained from analysis on the 0,1 scale but the standard errors are very different. These predicted means have been back-transformed by ASReml from the underlying (logistic) scale to the probability scale.

The initial analysis (on the 0,1 probability scale) ignores the variance differences associated with binomial data.

|                 |                 |                 |                 |
|-----------------|-----------------|-----------------|-----------------|
| PxP 1980        | BRxP 1980       | BxR 1980        | BRxP 1981       |
| 0.0162 ± 0.0067 | 0.0400 ± 0.0112 | 0.0659 ± 0.0285 | 0.0334 ± 0.0084 |

ASReml has an '*Analysis of Deviance*' option which we now demonstrate. In a mixed model, the variance components will change depending on which fixed terms are in the model. This will invalidate the Analysis of Deviance unless the variance components are fixed at the full model solution. So, fitting the model line

## 16.13 Multivariate animal genetics data – Sheep

---

```
Rot !BIN !TOT = Total !AODEV ~ mu SEX GRP SEX.GRP !r idv(SIRE 0.2555665 !GF)
```

produces the following Analysis of Deviance table

| Analysis of Deviance Table for Rot          |    |          |           |
|---|----|----------|-----------|
| Source of Variation                         | df | Deviance | Derived F |
| SEX   | 1  | 0.02     | 0.021     |
| GRP   | 3  | 4.42     | 1.848     |
| SEX.GRP                                     | 3  | 1.15     | 0.483     |
| Deviance from GLM fit                       | 48 | 38.27    |           |
| Variance heterogeneity factor [Deviance/DF] |    | 0.80     |           |

The Deviance value is the deviance calculated from the binomial part of the log-likelihood. This is distinct from the log-likelihood obtained by the REML algorithm which pertains to the working variable. Since the working variable changes with the model fitted, the LogL values are not comparable between models. The heterogeneity factor is the Deviance/df ratio and it gives some indication as to how well the discrete distribution describes the data. A value greater than 1 suggests the data is over-dispersed, that is the data values are more variable than expected under the chosen distribution.

In fitting generalized linear (mixed) models, the analysis is performed on a working variable predicted under the model and used to form weights for the analysis. The default is to scale these weights using a dispersion parameter value of 1 and is formally declared by writing the residual line as in any of the three forms presented below

```
residual idv(units !INIT 1 !GF)
residual id(units)
residual units
```

If there is significant over or under dispersion, it is sometimes desirable to scale the weights, typically by the *heterogeneity factor*. In this case, a fixed value of 0.80 so we can written as

```
residual idv(units !INIT 0.8 !GF)
```

Alternatively, we can let **ASReml** estimate the factor from the working variable by writing

```
residual idv(units)
```

in which case, for the present example, the heterogeneity factor is estimated as 0.5441.

**ASReml** solves for the linear effects twice (see the **!GLMM** qualifier) each iteration of the variance components so that the variance component updates are based on solutions obtained using the same variance parameters, *i.e.*, we start with a set of solutions and some parameters. We use these to update the solutions and use the updated solutions to update the variance parameter.

## 16.13 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 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.

### 16.13.1 Half-sib data 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  $\sigma_A^2$  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  $\sigma_m^2$  is the maternal variance component and the variance component for **litters** is denoted by  $\sigma_l^m$  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 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 !ARG 1 2 3 4 5 # A run for each trait
Multivariate Sire & Dam
tag
sire 92 !I
dam 3561 !I
grp 49
sex
brr 4
litter 4871
age
```

## 16.13 Multivariate animal genetics data – Sheep

```

wwt !M0 # !M0 identifies missing values
ywt !M0
gfw !M0
fdm !M0
fat !M0
coop.fmt !DOPART $1
!IF $1 == 1 !ASSIGN YV wwt # Sets up 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

!PART 1 2 3 5 # traits are substituted for $YV
$YV ~ mu age brr sex age.sex !r idv(sire) idv(dam) idv(lit) idv(age.grp),
idv(sex.grp) !f grp
residual idv(units)

!PART 4 # It 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
residual idv(units)

```

Table 16.13 and Table 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 16.13: REML estimates of a subset of the variance parameters for each trait (expressed as a ratio to their asymptotic standard error)

| term    | wwt  | ywt  | gfw  | fdm  | fat  |
|---------|------|------|------|------|------|
| sire    | 3.68 | 3.57 | 3.95 | 1.92 | 2.02 |
| dam     | 6.25 | 4.93 | 2.78 | 0.37 | 2.46 |
| litter  | 8.79 | 0.99 | 2.23 | 1.91 | 0.00 |
| age.grp | 2.30 | 1.39 | 0.31 | 1.15 | 1.66 |
| sex.grp | 2.90 | 3.43 | 3.70 | -    | 1.76 |

Table 16.14: Wald F statistics of the fixed effects for each trait for the genetic example

| term    | Wwt   | ywt   | gfw  | fdm | fat  |
|---------|-------|-------|------|-----|------|
| age     | 331.5 | 67.1  | 52.3 | 2.6 | 7.2  |
| brr     | 554.6 | 73.5  | 14.9 | 0.3 | 13.8 |
| sex     | 196.0 | 126.3 | 0.2  | 3.1 | 0.6  |
| age.sex | 10.3  | 1.7   | 1.9  | -   | 5.7  |

Thus, in the multivariate analysis we consider fitting the following models to the sire, dam and litter effects

$$\text{var}(\mathbf{u}_d) = \boldsymbol{\Sigma}_d \otimes \mathbf{I}_{3561}$$

$$\text{var}(\mathbf{u}_s) = \boldsymbol{\Sigma}_s \otimes \mathbf{I}_{92}$$

$$\text{var}(\mathbf{u}_l) = \boldsymbol{\Sigma}_l \otimes \mathbf{I}_{4891}$$

where  $\boldsymbol{\Sigma}_s^{5 \times 5}$ ,  $\boldsymbol{\Sigma}_d^{3 \times 3}$  and  $\boldsymbol{\Sigma}_l^{4 \times 4}$  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

$$\boldsymbol{\Sigma}_e \otimes \mathbf{I}_{7043}$$

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 !ARG 3 !WORKSPACE 10
Multivariate Sire & Dam
!DOPATH $1
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

!PATH 1
coop.fmt

!PATH 2
coop.fmt !CONTINUE coop_MV1.rsv # uses initial values from previous .rsv file

!PATH 3
coop.fmt !CONTINUE coop_MV2.rsv !EXTRA 3 !MAXIT 20

!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 57:66 for traits 1 2 3 5
F phen $uusT[1:6] + $susT[1:6] + $lusT[1:6] + Damv[1:6]
# Defines [1:6] elements of phen
# Defines 67:72 = 1:6 + 23:28 + 44:49 + 57:62
F phen $uusT[7:10] + $susT[7:10] + $lusT[7:10]
# Defines [7:10] elements of phen
# Defines 73:76 = 7:10 + 29:32 + 50:53
F phen $uusT[11:13] + $susT[11:13] + Damv[7:9]
F phen $uusT[14] + $susT[14]
F phen $uusT[15] + $susT[15] + Damv[10]
# Defines [11:15] elements of phen
# Defines 77:81 = 11:15 + 33:37
F Direct $susT *4. # Defines 82:96 = 23:37 * 4.
F Maternal Damv[1:6] - $susT[1:6] # Defines 97:102 = 54:59 - 23:28
F Maternal Damv[7:9] - $susT[11:13]
F Maternal Damv[10] - $susT[15]
F resid phen - $susT # Defines 107:121 = 67:81 - 23-37
H WWTh2 Direct[1] phen[1]
H YWTh2 Direct[3] phen[3]
H GFWh2 Direct[6] phen[6]
H FDMh2 Direct[10] phen[10]
H FATH2 Direct[15] phen[15]
R GenCor $susT
R MatCor Maternal

```

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       | $\Sigma_s$ | diag      | fa1       | us        |
| dam        | $\Sigma_d$ | diag      | fa1       | fa1       |
| litter     | $\Sigma_l$ | diag      | fa1       | us        |
| error      | $\Sigma_e$ | us        | us        | us        |
| LogL       |            | -21566.45 | -21488.11 | -21480.89 |
| 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 -21566.45. 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 -21488.11.

In this case the dam variance parameters are

```
xfal(TrDam123).id(dam)      14244 effects
TrDam123      XFA_V  0  1  0.00000      0.00000      0.00  0 F
TrDam123      XFA_V  0  2  0.00000      0.00000      0.00  0 F
TrDam123      XFA_V  0  3  0.800312E-02  0.800312E-02  1.55  0 P
TrDam123      XFA_L  1  1  1.47828      1.47828      13.96  0 P
TrDam123      XFA_L  1  2  1.70634      1.70634      11.66  0 P
TrDam123      XFA_L  1  3  0.135658     0.135658      8.34  0 P
```

And two of the dam specific variances is zero. The resulting dam matrix is

```
Covar/Var/Corr Matrix XFA xfal(TrDam123).id(dam)
2.187      1.000      0.8354      1.000
2.525      2.915      0.8354      1.000
0.2006     0.2317     0.2638E-01  0.8354
1.479      1.707      0.1357      1.000
```

On the basis of this **ASReml** with `!ARG 3`, fits unstructured matrices for sire and litter and `xfal` for dam using initial values derived from the previous analysis in `coopmf2.rsv`. Portions of the `.asr` file from the `Path 3` run are

Note: ReStartValues taken from `coop_MV2.rsv`

```
Note: Parameter constraint difference in .rsv file: P ==> F
Note: Use .msv/.tsv to reset parameter constraints.
* This job uses all of the 4 processor threads. *
Note: 76 singularities detected in design matrix.
1 LogL= -21488.8      S2=  1.0000      18085 df :  1 components restrained
2 LogL= -21484.7      S2=  1.0000      18085 df
3 LogL= -21481.8      S2=  1.0000      18085 df :  1 components restrained
4 LogL= -21481.0      S2=  1.0000      18085 df
5 LogL= -21480.9      S2=  1.0000      18085 df
6 LogL= -21480.9      S2=  1.0000      18085 df

- - - Results from analysis of wwt ywt gfw fdm fat - - -
Akaike Information Criterion      43063.77 (assuming 51 parameters).
Bayesian Information Criterion    43461.72
Note: Akaike parameter count excludes components bound at/near zero and
      is adjusted for any identifiability constraints on factor loadings.
```



## 16.13 Multivariate animal genetics data – Sheep

| Model_Term   |        |       |         | Sigma         | Sigma         | Sigma/SE | % C  |
|--|--------|-------|---------|---------------|---------------|----------|------|
| id(units) .us (Trait)                              |        | 35215 | effects |               |               |          |      |
| Trait  | US_V   | 1     | 1       | 9.46115       | 9.46115       | 33.29    | 0 P  |
| Trait  | US_C   | 2     | 1       | 7.34184       | 7.34184       | 20.55    | 0 P  |
| Trait  | US_V   | 2     | 2       | 17.6051       | 17.6051       | 27.09    | 0 P  |
| Trait  | US_C   | 3     | 1       | 0.272547      | 0.272547      | 8.38     | 0 P  |
| Trait  | US_C   | 3     | 2       | 0.668003      | 0.668003      | 13.99    | 0 P  |
| Trait  | US_V   | 3     | 3       | 0.141594      | 0.141594      | 23.70    | 0 P  |
| Trait  | US_C   | 4     | 1       | 0.962870      | 0.962870      | 2.89     | 0 P  |
| Trait  | US_C   | 4     | 2       | 1.99730       | 1.99730       | 3.64     | 0 P  |
| Trait  | US_C   | 4     | 3       | 0.286933      | 0.286933      | 5.08     | 0 P  |
| Trait  | US_V   | 4     | 4       | 3.64341       | 3.64341       | 9.00     | 0 P  |
| Trait  | US_C   | 5     | 1       | 0.850305      | 0.850305      | 8.48     | 0 P  |
| Trait  | US_C   | 5     | 2       | 2.48314       | 2.48314       | 19.33    | 0 P  |
| Trait  | US_C   | 5     | 3       | 0.786092E-01  | 0.786092E-01  | 7.04     | 0 P  |
| Trait  | US_C   | 5     | 4       | 0.115877      | 0.115877      | 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       | 1.01115       | 1.01115       | 2.97     | 0 P  |
| TrSG123  | DIAG_V | 2     | 1       | 16.0229       | 16.0229       | 3.50     | 0 P  |
| TrSG123  | DIAG_V | 3     | 1       | 0.280257      | 0.280257      | 3.71     | 0 P  |
| diag(TrAG1245) .age.grp                            |        | 196   | effects |               |               |          |      |
| TrAG1245   | DIAG_V | 1     | 1       | 0.132710E-02  | 0.132710E-02  | 2.01     | 0 P  |
| TrAG1245   | DIAG_V | 2     | 1       | 0.976527E-03  | 0.976527E-03  | 1.21     | 0 P  |
| TrAG1245   | DIAG_V | 3     | 1       | 0.176754E-02  | 0.176754E-02  | 1.13     | 0 P  |
| TrAG1245   | DIAG_V | 4     | 1       | 0.208072E-03  | 0.208072E-03  | 1.62     | 0 P  |
| us (Trait) .id (sire)                              |        | 460   | effects |               |               |          |      |
| Trait  | US_V   | 1     | 1       | 0.593939      | 0.593939      | 3.68     | 0 P  |
| Trait  | US_C   | 2     | 1       | 0.677371      | 0.677371      | 3.18     | 0 P  |
| Trait  | US_V   | 2     | 2       | 1.55634       | 1.55634       | 3.90     | 0 P  |
| Trait  | US_C   | 3     | 1       | 0.280540E-01  | 0.280540E-01  | 1.53     | 0 P  |
| Trait  | US_C   | 3     | 2       | 0.287728E-02  | 0.287728E-02  | 0.10     | 0 P  |
| Trait  | US_V   | 3     | 3       | 0.150181E-01  | 0.150181E-01  | 4.01     | 0 P  |
| Trait  | US_C   | 4     | 1       | 0.583800E-01  | 0.583800E-01  | 0.53     | 0 P  |
| Trait  | US_C   | 4     | 2       | -0.671299E-01 | -0.671299E-01 | -0.42    | 0 P  |
| Trait  | US_C   | 4     | 3       | 0.446406E-02  | 0.446406E-02  | 0.23     | 0 P  |
| Trait  | US_V   | 4     | 4       | 0.157536      | 0.157536      | 1.84     | 0 P  |
| Trait  | US_C   | 5     | 1       | 0.407081E-01  | 0.407081E-01  | 0.99     | 0 P  |
| Trait  | US_C   | 5     | 2       | 0.133348      | 0.133348      | 1.98     | 0 P  |
| Trait  | US_C   | 5     | 3       | 0.877985E-03  | 0.877985E-03  | 0.15     | 0 P  |
| Trait  | US_C   | 5     | 4       | -0.473699E-01 | -0.473699E-01 | -1.54    | 0 P  |
| Trait  | US_V   | 5     | 5       | 0.326711E-01  | 0.326711E-01  | 2.00     | 0 P  |
| xfal (TrDam123) .id (dam)                          |        | 14244 | effects |               |               |          |      |
| TrDam123   | XFA_V  | 0     | 1       | 0.00000       | 0.00000       | 0.00     | 0 F  |
| TrDam123   | XFA_V  | 0     | 2       | 0.00000       | 0.00000       | 0.00     | 0 F  |
| TrDam123   | XFA_V  | 0     | 3       | 0.661506E-02  | 0.661506E-02  | 1.25     | 0 P  |
| TrDam123   | XFA_L  | 1     | 1       | 1.46900       | 1.46900       | 12.81    | 0 P  |
| TrDam123   | XFA_L  | 1     | 2       | 1.51632       | 1.51632       | 7.99     | -1 P |
| TrDam123   | XFA_L  | 1     | 3       | 0.110527      | 0.110527      | 5.26     | -1 P |
| us (TrLit1234) .id (lit)                           |        | 19484 | effects |               |               |          |      |
| TrLit1234  | US_V   | 1     | 1       | 3.55307       | 3.55307       | 8.54     | 0 P  |
| TrLit1234  | US_C   | 2     | 1       | 1.53767       | 1.53767       | 3.33     | 0 P  |
| TrLit1234  | US_V   | 2     | 2       | 2.56347       | 2.56347       | 3.34     | 0 P  |
| TrLit1234  | US_C   | 3     | 1       | -0.311316E-01 | -0.311316E-01 | -0.74    | 0 P  |
| TrLit1234  | US_C   | 3     | 2       | 0.457652E-01  | 0.457652E-01  | 0.79     | 1 P  |
| TrLit1234  | US_V   | 3     | 3       | 0.191532E-01  | 0.191532E-01  | 2.47     | 0 P  |
| TrLit1234  | US_C   | 4     | 1       | -0.721744E-01 | -0.721744E-01 | -0.22    | 0 P  |
| TrLit1234  | US_C   | 4     | 2       | -0.793861     | -0.793861     | -1.55    | 0 P  |
| TrLit1234  | US_C   | 4     | 3       | -0.416784E-01 | -0.416784E-01 | -0.76    | 0 P  |
| TrLit1234  | US_V   | 4     | 4       | 0.897409      | 0.897409      | 2.29     | 0 P  |
| Covariance/Variance/Correlation Matrix US Residual |        |       |         |               |               |          |      |

## 16.13 Multivariate animal genetics data – Sheep

```

9.461      0.5689      0.2355      0.1640      0.2164
7.342      17.60       0.4231      0.2493      0.4633
0.2726     0.6681     0.1416     0.3994     0.1635
0.9628     1.997      0.2869     3.643      0.4752E-01
0.8503     2.483      0.7861E-01 0.1158      1.632
Covariance/Variance/Correlation Matrix US us(Trait).id(sire)
0.5938     0.7045     0.2970     0.1891     0.2921
0.6771     1.556     0.1871E-01 -0.1368     0.5913
0.2804E-01 0.2859E-02 0.1502E-01 0.8839E-01 0.3950E-01
0.5785E-01 -0.6773E-01 0.4300E-02 0.1576     -0.6629
0.4068E-01 0.1333     0.8749E-03 -0.4756E-01 0.3267E-01
Covar/Var/Corr Matrix XFA xfal(TrDam123).id(dam)
2.158      1.000      0.8062      1.000
2.231      2.306      0.8062      1.000
0.1627     0.1682     0.1888E-01 0.8062
1.469      1.519     0.1108      1.000
Covar/Var/Corr Matrix US us(TrLit1234).id(lit)
3.553      0.5091     -0.1209     -0.4051E-01
1.535      2.557     0.2041     -0.5239
-0.3151E-01 0.4510E-01 0.1910E-01 -0.3183
-0.7235E-01 -0.7938     -0.4167E-01 0.8977

                                Wald F statistics
Source of Variation            NumDF            F-inc
19 Trait.age                   5              100.13
20 Trait.brr                   15             116.73
21 Trait.sex                   5              77.96
23 Trait.age.sex               4              4.17

```

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

$$h^2 = \frac{\sigma_A^2}{\sigma_P^2}$$

where  $\sigma_P^2$  is the phenotypic variance and is given by

$$\sigma_P^2 = \sigma_s^2 + \sigma_d^2 + \sigma_l^2 + \sigma_e^2$$

recalling that

$$\sigma_s^2 = \frac{1}{4}\sigma_A^2$$

$$\sigma_d^2 = \frac{1}{4}\sigma_A^2 + \sigma_m^2$$

In the half-sib analysis we only use the estimate of additive genetic variance from the sire variance component. ASReml then reads out the VPREDICT instructions in the .asr file, stores the instructions in a .pin file and produces the following output in a .pvc file (edited).

```

:
X Damv xfal(TrDam123)
57 Damv                2.2602        0.33878
58 Damv                2.6039        0.39328
59 Damv                3.2133        0.67714
60 Damv                0.16422       0.31990E-01
61 Damv                0.20265       0.47685E-01
62 Damv                0.20063E-01 0.59498E-02
63 Damv                0.29626       0.82940E-01

```

## 16.13 Multivariate animal genetics data – Sheep

|   |              |             |
|---|--------------|-------------|
| 64 Damv   | 0.36560      | 0.11048     |
| 65 Damv   | 0.23057E-01  | 0.74499E-02 |
| 66 Damv   | 0.99139E-01  | 0.38068E-01 |
| F phen id(units).us(Trait);us(Trait)[1:6] + sire;us(Trait)[1:6] + lit;us(TrLit1 |              |             |
| 234)[1:6] + Damv[1:6]   |              |             |
| 67 phen   | 15.801       | 0.31400     |
| 68 phen   | 11.857       | 0.37837     |
| 69 phen   | 24.068       | 0.63636     |
| 70 phen   | 0.43346      | 0.33087E-01 |
| 71 phen   | 0.88706      | 0.44642E-01 |
| 72 phen   | 0.19474      | 0.54982E-02 |
| F phen id(units).us(Trait);us(Trait)[7:10] + sire;us(Trait)[7:10] + lit;us(TrLi |              |             |
| t1234)[7:10]  |              |             |
| 73 phen   | 0.94343      | 0.29839     |
| 74 phen   | 1.1256       | 0.37610     |
| 75 phen   | 0.25016      | 0.37295E-01 |
| 76 phen   | 4.6991       | 0.22561     |
| F phen id(units).us(Trait);us(Trait)[11:13] + sire;us(Trait)[11:13] + Damv[7:9] |              |             |
| 77 phen   | 0.97877      | 0.10902     |
| 78 phen   | 2.6172       | 0.14164     |
| 79 phen   | 0.84221E-01  | 0.12159E-01 |
| F phen id(units).us(Trait);us(Trait)[14] + sire;us(Trait)[14]                   |              |             |
| 80 phen   | 0.73786E-01  | 0.97200E-01 |
| F phen id(units).us(Trait);us(Trait)[15] + sire;us(Trait)[15] + Damv[10]        |              |             |
| 81 phen   | 1.5671       | 0.48495E-01 |
| F Direct sire;us(Trait) *4.   |              |             |
| 82 Direct   | 2.3551       | 0.64252     |
| 83 Direct   | 2.6465       | 0.84148     |
| 84 Direct   | 6.0692       | 1.5683      |
| 85 Direct   | 0.10781      | 0.72880E-01 |
| 86 Direct   | 0.21086E-02  | 0.10967     |
| 87 Direct   | 0.59620E-01  | 0.14911E-01 |
| 88 Direct   | 0.27240      | 0.44447     |
| 89 Direct   | -0.22136     | 0.64071     |
| 90 Direct   | 0.23698E-01  | 0.76898E-01 |
| 91 Direct   | 0.63658      | 0.34878     |
| 92 Direct   | 0.14588      | 0.16291     |
| 93 Direct   | 0.49004      | 0.26296     |
| 94 Direct   | -0.17483E-02 | 0.23578E-01 |
| 95 Direct   | -0.17798     | 0.12146     |
| 96 Direct   | 0.13072      | 0.63885E-01 |
| F Maternal Damv[1:6] - sire;us(Trait)[1:6]                                      |              |             |
| 97 Maternal   | 1.6714       | 0.37778     |
| 98 Maternal   | 1.9422       | 0.44909     |
| 99 Maternal   | 1.6960       | 0.78779     |
| 100 Maternal  | 0.13726      | 0.37030E-01 |
| 101 Maternal  | 0.20213      | 0.55244E-01 |
| 102 Maternal  | 0.51582E-02  | 0.70172E-02 |
| F Maternal Damv[7:9] - sire;us(Trait)[11:13]                                    |              |             |
| 103 Maternal  | 0.25979      | 0.93058E-01 |
| 104 Maternal  | 0.24309      | 0.12998     |
| 105 Maternal  | 0.23494E-01  | 0.95241E-02 |
| F Maternal Damv[10] - sire;us(Trait)[15]  |              |             |
| 106 Maternal  | 0.66459E-01  | 0.41358E-01 |
| F resid phen - sire;us(Trait)   |              |             |
| 107 resid   | 15.212       | 0.27763     |
| 108 resid   | 11.195       | 0.32255     |
| 109 resid   | 22.550       | 0.51565     |
| 110 resid   | 0.40651      | 0.28273E-01 |
| 111 resid   | 0.88653      | 0.36308E-01 |
| 112 resid   | 0.17984      | 0.41810E-02 |
| 113 resid   | 0.87533      | 0.28017     |
| 114 resid   | 1.1810       | 0.34629     |
| 115 resid   | 0.24423      | 0.32786E-01 |
| 116 resid   | 4.5400       | 0.21404     |
| 117 resid   | 0.94230      | 0.10220     |
| 118 resid   | 2.4946       | 0.12855     |

## 16.13 Multivariate animal genetics data – Sheep

```

119 resid                                0.84658E-01    0.10899E-01
120 resid                                0.11828          0.94394E-01
121 resid                                1.5344           0.46822E-01

H WWTh2 Direct[1] phen[1]
  WWTh2      = Direct      82/phen      67=          0.1491    0.0394
H YWTh2 Direct[3] phen[3]
  YWTh2      = Direct      84/phen      69=          0.2522    0.0615
H GFWh2 Direct[6] phen[6]
  GFWh2      = Direct      87/phen      72=          0.3061    0.0713
H FDMh2 Direct[10] phen[10]
  FDMh2      = Direct      91/phen      76=          0.1355    0.0724
H FATH2 Direct[15] phen[15]
  FATH2      = Direct      96/phen      81=          0.0834    0.0402
R GenCor sire;us(Trait)
  GenCor  2  1 = us(Tr 25/SQR[us(Tr 24*us(Tr 26)= 0.7000    0.1040
  GenCor  3  1 = us(Tr 27/SQR[us(Tr 24*us(Tr 29)= 0.2877    0.1734
  GenCor  3  2 = us(Tr 28/SQR[us(Tr 26*us(Tr 29)= 0.0035    0.1822
  GenCor  4  1 = us(Tr 30/SQR[us(Tr 24*us(Tr 33)= 0.2225    0.3486
  GenCor  4  2 = us(Tr 31/SQR[us(Tr 26*us(Tr 33)= -0.1126    0.3260
  GenCor  4  3 = us(Tr 32/SQR[us(Tr 29*us(Tr 33)= 0.1216    0.3860
  GenCor  5  1 = us(Tr 34/SQR[us(Tr 24*us(Tr 38)= 0.2629    0.2744
  GenCor  5  2 = us(Tr 35/SQR[us(Tr 26*us(Tr 38)= 0.5502    0.2076
  GenCor  5  3 = us(Tr 36/SQR[us(Tr 29*us(Tr 38)= -0.0198    0.2674
  GenCor  5  4 = us(Tr 37/SQR[us(Tr 33*us(Tr 38)= -0.6170    0.3856
R MatCor Maternal
  MatCor  2  1 = Mater 98/SQR[Mater 97*Mater 99]= 1.1536    0.1865
  MatCor  3  1 = Mater100/SQR[Mater 97*Mater102]= 1.4783    0.9060
  MatCor  3  2 = Mater101/SQR[Mater 99*Mater102]= 2.1610    1.4026
  MatCor  4  1 = Mater103/SQR[Mater 97*Mater106]= 0.7795    0.2918
  MatCor  4  2 = Mater104/SQR[Mater 99*Mater106]= 0.7241    0.3109
  MatCor  4  3 = Mater105/SQR[Mater102*Mater106]= 1.2689    0.9344
Note: The parameter estimates are followed by
      their approximate standard errors.

```

### 16.13.2 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 three fields of the data file `pcoop.fmt` and no historical relationships are available for this data so the data file doubles as the pedigree file.

The multi-trait additive genetic variance matrix,  $\Sigma_A$  of the animals (sires, dams and lambs) is given by

$$\text{var}(\mathbf{u}_A) = \Sigma_A \otimes \mathbf{A}$$

where  $\mathbf{A}$  is the genetic relationship matrix and  $\mathbf{u}_A$  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

## 16.13 Multivariate animal genetics data – Sheep

---

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 !ARG 2 3 4 5 !WORKSPACE 10
Multivariate Animal Model
!DOPATH $1
  tag !P
  sire 92 !I
  dam !P
  grp 49
  sex
  brr 4
  litter 4871
  age
  wwt !M0 # !M0 identifies missing values
  ywt !M0
  gfw !M0
  fdm !M0
  fat !M0
pcoop.fmt !EXTRA 6
!DOPART $1 # Read pedigree from first three fields

!PATH 1 // pcoop.fmt
!PATH 2 // pcoop.fmt !CONTINUE coop_MVAnimal1.rsv !MAXIT 40
!PATH 3 // pcoop.fmt !CONTINUE coop_MVAnimal2.rsv !MAXIT 40
!PATH 4 // pcoop.fmt !CONTINUE coop_MVAnimal2.rsv !MAXIT 40
!PATH 5 // pcoop.fmt !CONTINUE coop_MVAnimal4.rsv !MAXIT 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 spacifications 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,
xfal(Trait).nrm(tag) xfal(TrDam12).nrm(dam) xfal(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) xfal(TrDam12).nrm(dam) us(TrLit1234).id(lit),
!f Trait.grp
residual id(units).us(Trait)
```

```

!PATH 4
wgt 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
wgt 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 in 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<sup>1</sup> since dam is defined with !P. In this case there is no difference between fitting `nrm(dam)` and `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 `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.

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

Note: ReStartValues taken from coop\_MVAnimal4.rsv

```

Note: Parameter constraint difference in .rsv file: P ==> F
Note: Use .msv/.tsv to reset parameter constraints.

```

```

Note: The XFA term in xfa3(Trait).nrm(tag) appears to have more free parameters (20-3)
      than formally expected (15), ignoring any formal user constraints.
      By default ASReml uses a Levenberg type damping modification when
      calculating updates for loadings. This makes the process more robust
      and avoids singularities in the AI matrix. However, while the
      likelihood is maximised, the FA parameters obtained are not unique.
      Use !AIPENALTY 0 to avoid the Levenberg type damping modification.
* This job uses all of the 4 processor threads. *

```

```

Note: XFA model: lower loadings initially held fixed.
Note: 76 singularities detected in design matrix.
 1 LogL= -21483.4      S2=  1.0000      18085 df :   1 components restrained
 2 LogL= -21482.7      S2=  1.0000      18085 df :   1 components restrained

Note: XFA model fitted with rotation.
 3 LogL= -21482.1      S2=  1.0000      18085 df :   1 components restrained
 4 LogL= -21482.0      S2=  1.0000      18085 df
 5 LogL= -21482.0      S2=  1.0000      18085 df
 6 LogL= -21481.9      S2=  1.0000      18085 df
 7 LogL= -21481.9      S2=  1.0000      18085 df
 8 LogL= -21481.9      S2=  1.0000      18085 df

```

## 16.13 Multivariate animal genetics data – Sheep

- - - Results from analysis of wwt ywt gfw fdm fat - - -  
Akaike Information Criterion 43063.85 (assuming 50 parameters).  
Bayesian Information Criterion 43453.99  
Note: Akaike parameter count excludes components bound at/near zero and  
is adjusted for any identifiability constraints on factor loadings.

| Model_Term             |        |       |         | Sigma         | Sigma         | Sigma/SE | % C  |
|------------------------|--------|-------|---------|---------------|---------------|----------|------|
| id(units).us(Trait)    |        | 35200 | effects |               |               |          |      |
| Trait                  | US_V   | 1     | 1       | 7.52821       | 7.52821       | 17.46    | -1 P |
| Trait                  | US_C   | 2     | 1       | 4.88663       | 4.88663       | 8.47     | 0 P  |
| Trait                  | US_V   | 2     | 2       | 13.0992       | 13.0992       | 12.17    | 0 P  |
| Trait                  | US_C   | 3     | 1       | 0.119659      | 0.119659      | 2.37     | 0 P  |
| Trait                  | US_C   | 3     | 2       | 0.497960      | 0.497960      | 6.55     | 0 P  |
| Trait                  | US_V   | 3     | 3       | 0.105727      | 0.105727      | 11.38    | 0 P  |
| Trait                  | US_C   | 4     | 1       | 0.943944      | 0.943944      | 2.26     | 0 P  |
| Trait                  | US_C   | 4     | 2       | 2.23817       | 2.23817       | 3.41     | 0 P  |
| Trait                  | US_C   | 4     | 3       | 0.290907      | 0.290907      | 4.34     | 1 P  |
| Trait                  | US_V   | 4     | 4       | 3.35174       | 3.35174       | 7.72     | 0 P  |
| Trait                  | US_C   | 5     | 1       | 0.465954      | 0.465954      | 2.94     | 0 P  |
| Trait                  | US_C   | 5     | 2       | 1.74913       | 1.74913       | 7.43     | 0 P  |
| Trait                  | US_C   | 5     | 3       | 0.488729E-01  | 0.488729E-01  | 2.39     | 0 P  |
| Trait                  | US_C   | 5     | 4       | 0.252839      | 0.252839      | 1.75     | 0 P  |
| Trait                  | US_V   | 5     | 5       | 1.44358       | 1.44358       | 19.21    | 0 P  |
| diag(TrSG123).sex.grp  |        | 147   | effects |               |               |          |      |
| TrSG123                | DIAG_V | 1     |         | 0.994883      | 0.994883      | 2.95     | 0 P  |
| TrSG123                | DIAG_V | 2     |         | 15.9950       | 15.9950       | 3.51     | 0 P  |
| TrSG123                | DIAG_V | 3     |         | 0.281451      | 0.281451      | 3.71     | 0 P  |
| diag(TrAG1245).age.grp |        | 196   | effects |               |               |          |      |
| TrAG1245               | DIAG_V | 1     |         | 0.132361E-02  | 0.132361E-02  | 2.01     | 0 P  |
| TrAG1245               | DIAG_V | 2     |         | 0.915943E-03  | 0.915943E-03  | 1.16     | 0 P  |
| TrAG1245               | DIAG_V | 3     |         | 0.175676E-02  | 0.175676E-02  | 1.13     | 0 P  |
| TrAG1245               | DIAG_V | 4     |         | 0.219379E-03  | 0.219379E-03  | 1.66     | 0 P  |
| us(TrLit1234).id(lit)  |        | 19484 | effects |               |               |          |      |
| TrLit1234              | US_V   | 1     | 1       | 3.71446       | 3.71446       | 8.86     | 0 P  |
| TrLit1234              | US_C   | 2     | 1       | 2.00118       | 2.00118       | 4.34     | 0 P  |
| TrLit1234              | US_V   | 2     | 2       | 2.70323       | 2.70323       | 3.86     | 0 P  |
| TrLit1234              | US_C   | 3     | 1       | 0.457548E-01  | 0.457548E-01  | 1.24     | 0 P  |
| TrLit1234              | US_C   | 3     | 2       | 0.133243      | 0.133243      | 2.63     | 0 P  |
| TrLit1234              | US_V   | 3     | 3       | 0.207632E-01  | 0.207632E-01  | 3.15     | 0 P  |
| TrLit1234              | US_C   | 4     | 1       | -0.100612     | -0.100612     | -0.29    | 0 P  |
| TrLit1234              | US_C   | 4     | 2       | -0.740633     | -0.740633     | -1.39    | 0 P  |
| TrLit1234              | US_C   | 4     | 3       | -0.402453E-01 | -0.402453E-01 | -0.69    | 0 P  |
| TrLit1234              | US_V   | 4     | 4       | 0.739304      | 0.739304      | 1.83     | 0 P  |
| xfal(TrDam12).nrm(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       | 1.01119       | 1.01119       | 5.76     | 0 P  |
| TrDam12                | XFA_L  | 1     | 2       | 0.641762      | 0.641762      | 2.03     | 0 P  |
| dam                    | NRM    |       | 10696   |               |               |          |      |
| xfa3(Trait).nrm(tag)   |        | 85568 | effects |               |               |          |      |
| Trait                  | XFA_V  | 0     | 1       | 0.00000       | 0.00000       | 0.00     | 0 B  |
| Trait                  | XFA_V  | 0     | 2       | 1.38887       | 1.38887       | 0.62     | -1 P |
| Trait                  | XFA_V  | 0     | 3       | 0.381920E-01  | 0.381920E-01  | 0.78     | 0 P  |
| Trait                  | XFA_V  | 0     | 4       | 0.189163      | 0.189163      | 0.22     | -2 P |
| Trait                  | XFA_V  | 0     | 5       | 0.399278E-01  | 0.399278E-01  | 0.28     | 1 P  |
| Trait                  | XFA_L  | 1     | 1       | 1.86601       | 1.86601       | 0.00     | 0 F  |
| Trait                  | XFA_L  | 1     | 2       | 2.60859       | 2.60859       | 4.66     | 1 P  |
| Trait                  | XFA_L  | 1     | 3       | 0.134170      | 0.134170      | 1.85     | 0 P  |
| Trait                  | XFA_L  | 1     | 4       | -0.104914     | -0.104914     | -0.34    | -1 P |
| Trait                  | XFA_L  | 1     | 5       | 0.367346      | 0.367346      | 3.85     | 0 P  |
| Trait                  | XFA_L  | 2     | 1       | -0.517456     | -0.517456     | 0.00     | 0 F  |
| Trait                  | XFA_L  | 2     | 2       | 0.318167      | 0.318167      | 0.00     | 0 F  |
| Trait                  | XFA_L  | 2     | 3       | -0.553415E-01 | -0.553415E-01 | -0.36    | 1 P  |
| Trait                  | XFA_L  | 2     | 4       | -0.585559     | -0.585559     | -0.87    | -2 P |

## 16.13 Multivariate animal genetics data – Sheep

```

Trait          XFA_L  2  5  0.235830      0.235830      1.18  -1 P
Trait          XFA_L  3  1 -0.216978     -0.216978     -0.13  -1 P
Trait          XFA_L  3  2  0.159651      0.159651      0.08   0 P
Trait          XFA_L  3  3 -0.105749     -0.105749     -0.56   0 P
Trait          XFA_L  3  4  0.249561      0.249561      0.36   0 P
Trait          XFA_L  3  5 -0.512913E-01 -0.512913E-01 -0.12   1 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/Varianace/Correlation Matrix US Residual
  7.528      0.4921      0.1341      0.1879      0.1413
  4.887      13.10      0.4231      0.3378      0.4022
  0.1197     0.4980     0.1057      0.4887      0.1251
  0.9439     2.238     0.2909      3.352      0.1149
  0.4660     1.749     0.4887E-01  0.2528     1.444
Covar/Var/Corr Matrix US us(TrLit1234).id(lit)
  3.715      0.6317      0.1654     -0.6133E-01
  2.002      2.704      0.5624     -0.5261
  0.4594E-01  0.1333     0.2076E-01 -0.3290
 -0.1016     -0.7437    -0.4076E-01  0.7391

Covar/Var/Corr Matrix XFA xfa1(TrDam12).nrm(dam)
  1.025      1.000      1.000
  0.6506     0.4129      1.000
  1.013      0.6426      1.000
Covar/Var/Corr Matrix XFA xfa3(Trait).nrm(tag)
  3.785      0.8303      0.5832      0.3929E-01  0.6103      0.9591      -0.2660      -0.9711E-01
  4.657      8.310      0.4118     -0.1815      0.7354      0.9019      0.1104      0.5504E-01
  0.3013     0.3152     0.7048E-01 -0.2957E-01  0.3240      0.5075     -0.2194     -0.3930
  0.5947E-01 -0.4070     -0.6107E-02  0.6051     -0.5009     -0.1307     -0.7346      0.3167
  0.5735      1.024     0.4154E-01 -0.1882      0.2333      0.7619      0.4958     -0.1173
  1.866      2.600      0.1347     -0.1017      0.3680      1.000      0.000      0.000
 -0.5175     0.3182     -0.5826E-01 -0.5714      0.2395      0.000      1.000      0.000
 -0.1889     0.1587     -0.1043      0.2463     -0.5664E-01  0.000      0.000      1.000

Wald F statistics
Source of Variation      NumDF      F-inc
20 Trait.age              5          99.38
21 Trait.brr              15         116.33
22 Trait.sex              5          78.62
24 Trait.age.sex          4           4.14

```

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.
- Cullis, B. R., Smith, A. B. and Thompson, R. (2004). Perspectives of ANOVA, REML and a General linear mixed model, *In: Methods and Models in Statistics in honour of Professor John Nelder FRS*. Editor: Adams, N. M., Crowder, M. J., Hand, D. J. and Stephens, D. A. 53-94.
- Cullis, B. R., Smith, A. B. and Coombes, N. E. (2006). On the design of early generation variety trials with correlated data, *Journal of Agricultural, Biological and Environmental Statistics* **11**: 381-393.
- Dempster, A. P., Selwyn, M. R., Patel, C. M. and Roth, A. J. (1984). Statistical and computational aspects of mixed model analysis, *Applied Statistics* **33**: 203–214.
- Diggle, P. J., Ribeiro, P. J. Jr. and Christensen, O. F. (2003). An Introduction to Model-Based Geostatistics, *In: Spatial Statistics and Computational Methods*. Editor: Moller, J. Springer-Verlag, New York, 43-86.
- 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. (2019). Average information residual maximum likelihood in practice, *Journal of Animal Breeding and Genetics* **136**(4): 262-272.
- Gilmour, A. R., Anderson, R. D. and Rae, A. L. (1985). The analysis of binomial data by a

- generalised linear mixed model, *Biometrika* **72**: 593-599.
- Gilmour, A. R., Cullis, B. R. and Verbyla, A. P. (1997). Accounting for natural and extraneous variation in the analysis of field experiments, *Journal of Agricultural, Biological, and Environmental Statistics* **2**: 269–273.
- Gilmour, A. R., Cullis, B. R., Welham, S. J., Gogel, B. J. and Thompson, R. (2004). An efficient computing strategy for prediction in mixed linear models, *Computational Statistics and Data Analysis* **44**: 571–586.
- Gilmour, A. R., 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.
- Henderson, C. R., Kempthorne, O., Searle, S.R. and von Krosigk, C.M. (1959). The Estimation of environmental and genetic trends from records subject to culling, *Biometrics* **15**: 192-218.
- Kamman, 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). Small sample inference for 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 prediction, *Biometrics* **38**: 613–621.
- Lee, S. H. and van der Werf, J. H. J. (2016). MTG2: an efficient algorithm for multivariate linear mixed model analysis based on genomic information. *Bioinformatics* **32**(9): 1420-1422.

- Legarra, A., Aguilar, I. and Misztal, I. (2009). A relationship matrix including full pedigree and genomic information. *Journal of Dairy Science* **92**(9): 4656–4663.
- Martini, J. W., Schrauf, M. F., Garcia-Baccino, C. A., Pimentel, E. C., Munilla, S., Rogberg-Muñoz, A., Cantet, R. J., Reimer, C., Gao, N., Wimmer, V., and Simianer, H. (2018). The effect of the  $H^{-1}$  scaling factors  $\tau$  and  $\omega$  on the structure of  $H$  in the single-step procedure. *Genetics Selection Evolution* **50**(1): 1–9.
- Mazur, L. (2021). Computational methods for the fitting of factor analytic linear mixed models with applications to plant variety trials. PhD thesis, University of Wollongong, Australia.
- McCulloch, C. and Searle, S. R. (2001). *Generalized, Linear, and Mixed Models*, Wiley.
- Meuwissen, T.H.E. and Lou, Z. (1992). Computing inbreeding coefficients in large populations, *Genetics, Selection and Evolution* **24**: 305.
- Millar, R. and Willis, T. (1999). Estimating the relative density of snapper in and around a marine reserve using a log-linear mixed-effects model, *Australian and New Zealand Journal of Statistics* **41**: 383–394.
- Nelder, J. A. 1977. Reformulation of linear-models - a reformulation of linear model (with discussion), *Journal of the Royal Statistical Society, Series A (Statistics in Society)* **140**(1): 48–77.
- 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, M., Iwaisaki, H., and Colleau, J. J. (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): 719–727.
- 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 Liang, K. -Y. (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., 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 Mutilze, 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.
- Vitezica, Z. G., Legarra, A., Toro, M. A., and Varona, L. (2017). Orthogonal estimates of variances for additive, dominance, and epistatic effects in populations, *Genetics* **206**(3): 1297-1307.
- Waddington, D., Welham, S. J., Gilmour, A. R. and Thompson, R. (1994). Comparisons of some GLMM estimators for a simple binomial model, *Genstat Newsletter* **30**: 13–24.
- Webster, R. and Oliver, M. (2001) *Geostatistics for Environmental Scientists*, New York: John Wiley & Sons, Inc.
- 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.
- Wolfinger, R., Tobias, R. and Sall, J. (1994). Computing Gaussian Likelihoods and Their Derivatives for General Linear Mixed Models, *SIAM Journal on Scientific Computing* **15**(6): 1294-1310.
- Yates, F. (1935). Complex experiments, *Journal of the Royal Statistical Society, Series B* **2**: 181–247.

# Index

## A

ABORTASR.NOW, 58  
Access database, 36  
accuracy - genetic BLUP, 220  
advanced processing arguments, 194  
AI algorithm, 12  
ainverse.bin, 146  
Akaike Information Criteria (AIC), 15  
aliasing, 93, 94  
Analysis of Deviance, 90  
Analysis of Variance, 17  
animal breeding, 1  
animal model, 144, 297  
arguments, 3  
asrdata.bin, 70  
ASReml symbols  
  !{, 78  
  !}, 78  
  #, 35  
  \$, 35  
  \*, 35, 78  
  ,, 78  
  ., 35  
  /, 78  
  :, 77  
  ~, 76  
  +, 78  
associated factors, 83  
autoregressive, 107  
Average Information, 1

## B

balanced repeated measures, 268  
Bayesian Information Criteria(BIC), 15  
binary files, 36  
binomial divisor, 89  
BLUE, 13  
BLUP, 13

## C

combining variance models, 10  
command file, 25  
  genetic analysis, 144  
  multivariate, 142  
command line options, 188  
  A ASK, 189  
  B BRIEF, 190  
  C CONTINUE, 193  
  D DEBUG, 191  
  F FINAL, 193  
  Gg GRAPHICS, 191  
  Hg HARDCOPY, 191  
  I INTERACTIVE, 191  
  N NOGRAPHICS, 191  
  O ONERUN, 193  
  Q QUIET, 191  
  R RENAME, 193  
  X XML, 191

  Y YVAR, 193  
conditional distribution, 11  
conditional F statistics, 17  
conditional factors, 81  
convergence criterion, 58  
correlated effects, 14  
correlation, 209  
  between traits, 141  
  model, 9  
covariance model, 10  
covariates, 34, 51, 92  
cubic spline, 78, 309

## D

data field syntax, 38  
data file, 23, 34  
  binary format, 36  
  fixed format, 36  
  free format, 34  
  merging, 203  
  preparation, 34  
  using Excel, 36  
data file line, 27, 52  
  qualifiers, 52  
  syntax, 52  
data sets, 259  
  barley.asd, 275  
  grass.asd, 268  
  harvey.dat, 144  
  nin89.asd, 23  
  oats.asd, 259  
  orange.asd, 309  
  rat.dat, 141  
  rats.asd, 262  
  rice.asd, 299  
  voltage.asd, 266  
  wheat.asd, 282  
debug mode, 191  
Denominator Degrees of Freedom, 17  
dense, 93  
  vs sparse, 93  
design factors, 93  
diagnostics, 15  
diallel analysis, 83  
direct products, 8  
dispersion parameter, 90  
distribution, 11  
  conditional, 11  
  marginal, 11  
DOPATH, 188, 195

## E

Ecode, 33  
Eigen analysis, 237  
EM update, 116  
environment variable job control, 56  
equations  
  mixed model, 13  
errors, 240  
  common problems, 239

# Index

---

error messages, 239  
Excel, 36  
execution time, 199, 237

## F

F statistic, 17  
F statistics  
  incremental, 17  
  Wald, 17  
factor, 35  
  labels, 35  
factor levels, 42  
factor qualifier, 39  
  DATE, 41  
  DMY, 41  
  label, 41  
  LL Label Length, 41  
  MDY, 41  
  PRUNE, 41  
  PRUNEALL, 41  
  PRUNEOFF, 41  
  SORT, 42  
  SORTALL, 42  
  TIME, 41  
FINALASR.NOW, 58  
Fisher-scoring algorithm, 12  
fixed effects, 4, 14, 75, 96  
fixed format files, 53  
fixed terms, 76  
  in the model, 80  
  multivariate, 142  
  primary, 80  
  sparse, 80  
free format, 34  
functions of variance components, 32, 205  
  convert CORUH and XFA to US, 209  
  correlation, 209  
  linear combinations, 207  
  syntax, 205

## G

gamma distribution, 89  
generalized Linear Models, 88  
genetic  
  data, i  
  groups, 147  
  links, 144  
  markers, 60  
  models, 145  
  qualifiers, 145  
  relationships, 145  
GIV, 140  
GIV file, 147, 151  
GIV matrix, 151  
GLM distribution  
  Binomial, 88  
  Gamma, 89  
  Negative binomial, 90  
  Normal, 88  
  Ordinal data, 89  
  Poisson, 89  
GLMM, 91  
Graphics command line options, 191  
GRM, 140

## H

half-sib analysis, 324  
heritability, 237  
  calculating, 208  
hypothesis testing, 17

## I

IID, 4, 8, 28, 96  
inbreeding coefficients, 147, 220  
Incremental F statistics, 17  
information criterion, 15  
information matrix, 11, 12  
  expected, 12  
  observed, 11  
input file extensions, 35  
  .bin, 36  
  .csv, 35  
  .dbl, 36  
  .pin, 214  
interactions, 81

## J

job control, 52  
  options, 193  
  qualifiers, 52

## K

key output files, 215

## L

likelihood  
  convergence, 58  
  log residual, 11  
  offset, 218  
  ratio methods, 17  
  residual, 11  
longitudinal data, i, 309  
  balanced example, 309

## M

marginal distribution, 11  
Matérn variance structure, 138  
matrices  
  AR1, 103  
  correlation, 9  
  derivative, 115  
  design, 60, 75  
  expected information, 12  
  GIV, 140, 153  
  GRM, 140  
  identity, 5  
  non-singular, 129  
  NRM, 140  
  observed information, 11  
  variance, 6, 96  
measurement error, 108  
  nugget effect, 278  
MERGE  
  directive, 203  
  syntax, 203  
MET (multi-environment trial), 6, 114, 132

# Index

---

meta analysis, 1, 178  
missing values, 35, 92, 203, 204  
    in the response, 92  
    NA, 35  
mixed model, 4  
    effects, 4  
    equations, 13  
    multivariate, 141  
    specifying terms, 28  
model building, 314  
models  
    animal, 144, 297  
    correlation, 9  
    covariance, 10  
    formulae, 75  
    sire, 144  
moving average, 85, 136  
multi-environment trials, 6, 114, 132  
multivariate analysis, 141, 143, 324  
    example, 323  
    half-sib, 324

## N

Nebraska Intrastate Nursery, 22  
negative binomial, 88, 90, 91

## O

objective function, 12  
operators, 77  
options  
    command line, 188  
ordering of terms, 93  
ordinal data, 89  
orthogonal polynomials, 79, 174  
outliers, 238, 264  
output  
    files, 29, 215  
    objects, 237  
output file extensions, 215  
    .aov, 216, 222  
    .apj, 216  
    .ask, 216  
    .asl, 216, 224  
    .asp, 216  
    .asr, 215, 217  
    .ass, 216  
    .bgiv, 151, 152, 158, 160  
    .bgrm, 151, 152, 158, 160  
    .dbr, 216  
    .dgiv, 152, 158  
    .dgrm, 151, 152, 158  
    .dpr, 216, 225  
    .giv, 151, 158  
    .grm, 151, 158  
    .mef, 163, 166  
    .msv, 215, 225  
    .pvc, 215, 225  
    .pvs, 215, 226  
    .res, 215, 226  
    .rgiv, 151, 152, 158, 160  
    .rgrm, 151, 152, 158, 160  
    .rsv, 215, 233  
    .sgiv, 151, 152, 158, 160  
    .sgrm, 151, 152, 158, 160  
    .sln, 215, 220

.spr, 216  
.tab, 215, 234  
.tsv, 215, 234  
.veo, 216  
.vll, 216  
.vpc, 216, 225  
.vpv, 216, 235  
.vrb, 74, 216, 235  
.vvp, 216, 236  
.was, 216  
.wvr, 216, 236  
.xml, 216  
.yht, 216  
own models, 115  
OWN variance model, 69  
own variance structure, 115  
OWN variance structure  
    !F1, 115  
    !T, 115

## P

path  
    DOPATH, 187, 197  
    PATH, 187, 197  
pedigree file, 145  
pedigree modification parameters, 149  
performance issues, 199  
power, 238  
    model, 114, 131, 256, 271  
    structure, 256  
Predict  
    !PRWTS, 183  
    !TURNINGPOINTS, 178  
    \$TP, 86  
    PLOT options, 178  
predicted values, 30, 32  
prediction, 29, 172  
    estimable, 33  
    two-way interaction effects, 186

## Q

qualifier types  
    data file, 52  
    data input and output, 52  
    high-level, 195  
    job control, 56, 59, 64  
    pedigree modification, 151  
    variance model functions, 111  
qualifiers, 44  
    ! -, 45  
    ! \*, 45  
    ! +, 45  
    ! <, 46  
    ! <=, 46  
    ! <>, 46  
    ! =, 45  
    ! =, 46  
    ! >, 46  
    ! >=, 46  
    !/, 45  
    !^, 45  
    !A, 27, 39  
    !ABS, 46  
    !ADD, 152  
    !ADDd, 155



# Index

---

!ADJUST, 66  
!AIF, 147  
!AILOADINGS, 64  
!AIPENALTY, 64  
!AISINGULARITIES, 64  
!ALPHA, 147  
!AOD analysis of deviance, 90  
!ARCSIN, 46  
!ARGS, 190  
!ARLIMIT, 64  
!AS, 40  
!ASAVERAGE, 176  
!ASK, 190  
!ASMV, 59  
!ASSIGN, 195  
!ASSOCIATE factors, 180  
!ASSOCIATE prediction, 176  
!ASUV, 59  
!AVERAGE, 176  
!BINOMIAL GLM, 89  
!BLUP, 65  
!BMP, 65  
!BRIEF, 65, 190  
!CAMAT, 72  
!CENTRE, 163  
!CHECK, 204  
!CIMAT, 72  
!CINV, 72  
!CIPMAT, 72  
!CMAT, 72  
!COLFAC, 53  
!COLUMNFACTOR, 53  
!COMPLOGLOG, 88  
!CONTINUE, 56, 189  
!CONTRAST, 57  
!COORD, 113, 114  
!COS, 46  
!CPMAT, 72  
!CSKIP, 42, 147, 162  
!CSV, 53  
!CYCLE, 196, 199  
!D, 46  
!DATAFILE, 53  
!DDF, 57  
!DEBUG, 190  
!DEC, 177  
!DEFINE, 57, 214  
!DENSE, 65  
!DESIGN, 59  
!DEVIANCE residuals, 90  
!DF, 65  
!DIAG, 147  
!DISP dispersion, 90  
!DISPLAY, 59  
!DO, 46  
!DOM, 46  
!DOMINANCE, 162, 167  
!DOPART, 197  
!DOPATH, 197  
!DOUBLE, 72  
!DV, 46  
!EIGTRANSFORM, 152, 157  
!EMFLAG, 66  
!ENDDO, 46  
!EPISTATIC, 162, 167  
!EPS, 59  
!EQORDER, 66  
!ESTIMATE, 177, 185  
!EXCEPT, 177  
!EXCLUDE, 53  
!EXP, 46  
!EXTRA, 67  
!FACPOINTS, 73  
!FACTOR, 60  
!FCON, 20, 57  
!FGEN, 147  
!Fi, 113, 115  
!FIELD, 61  
!FILTER, 53  
!FINAL, 189  
!FOLDER, 53  
!FOR, 197  
!FORMAT, 53  
!FOWN, 20, 67  
!FREEGH, 68  
!G, 40, 58, 59  
!GAMMA GLM, 89  
!GDENSE, 68  
!GINDEX, 68  
!GKRIGE, 59  
!GLMM, 68  
!GOFFSET, 147  
!GRAPHICS, 190  
!GROUPFACTOR, 60  
!GROUPS, 147, 150  
!GROUPSDF, 152, 153  
!Gs, 113, 116  
!GSCALE, 163  
!HARDCOPY, 190  
!HINV, 152, 156  
!HOLD, 69  
!HPGL, 69  
!HSKIP, 152  
!I, 40  
!IDENTITY link, 88  
!IDLIMIT, 60  
!IF, 198  
!IGNORE, 177  
!INBRED, 148  
!INCLUDE, 56  
!INIT, 113, 117  
!INTERACTIVE, 190  
!INVERSE link, 88  
!Jddm, 47  
!Jmmd, 47  
!JOIN, 58, 60  
!Jyyd, 47  
!KEEP, 150, 204  
!KEEPGRM, 152, 156  
!KEY, 61, 204  
!KNOTS, 73  
!L, 39, 40  
!LAST, 69, 147  
!LDET, 152, 155  
!LINK, 119  
!LOGARITHM link, 88  
!LOGFILE, 189  
!LOGIT, 88  
!LOGIT link, 88  
!LONGINTEGER, 148  
!M, 47  
!MAKE, 148  
!MATCH, 55  
!MAX, 47  
!MAXIT, 58

# Index

---

!MBF, 60  
!MERGE, 55  
!MEUWISSEN, 148  
!MGS, 148  
!MIN, 47  
!MM, 47, 49  
!MOD, 47  
!MP, 61  
!MSV, 56  
!MULTINOMIAL GLM, 89  
!MVINCLUDE, 61  
!MVREMOVE, 61  
!NA, 47  
!ND, 152  
!NEGBIN GLM, 89  
!NOCHECK, 73  
!NODISPLAY, 61  
!NODUP, 204  
!NOGRAPHS, 189  
!NOID, 162  
!NOKEY, 61  
!NONAMES, 162  
!NONULL, 152, 155  
!NOPRUNE, 150  
!NOREORDER, 73  
!NORMAL, 47  
!NORMAL GLM, 88  
!NOSCRATCH, 73  
!NOZEROVC, 69  
!NSD, 152  
!OFFSET variable, 90  
!OMEGA, 152  
!ONERUN, 190  
!ONLYUSE, 177  
!operator, 44  
!OUTFOLDER, 190  
!OUTLIER, 15, 69  
!OWN, 69  
!P, 40  
!PAC, 177  
!PARALLEL, 176  
!PATH, 198  
!PEARSON residuals, 90  
!PEV, 163  
!PLOT, 177  
!PNG, 69  
!POISSON GLM, 89  
!POLPOINTS, 74  
!PPOINTS, 74  
!PRECISION, 152, 155  
!PRESENT, 176  
!PRINT, 69  
!PRINTALL, 172, 178  
!PROBIT link, 88  
!PRWTS, 176  
!PS, 69  
!PSD, 152, 163  
!PVAL, 59, 61  
!PVR GLM fitted values, 90  
!PVIFORM, 70  
!PVW GLM fitted values, 90  
!PXEM, 66  
!QUAAS, 148  
!QUIET, 190  
!RANGE, 163  
!READ, 55  
!RECODE, 55  
!REDUCE, 150  
!RENAME, 60, 190  
!REPEAT, 148  
!REPLACE, 47  
!REPORT, 74  
!RESCALE, 47  
!RESIDUALS, 70  
!RESPONSE residuals, 91  
!RFIELD, 61  
!ROWFAC, 55  
!ROWFACTOR, 55  
!RREC, 55  
!RSKIP, 55  
!S2LIMIT, 70  
!SAMEDATA, 196  
!SARGOLZAEI, 148  
!SAVE, 70  
!SAVEGIV, 148, 152, 155, 163  
!SCALE, 74, 119  
!SCORE, 74  
!SCREEN, 70  
!SECTION, 62  
!SED, 178  
!SEED, 47  
!SELECT, 53  
!SELF, 148  
!SET, 47  
!SETN, 47  
!SETU, 48  
!SIGMAP, 107  
!SIN, 46  
!SKIP, 35, 42, 53, 55, 56, 61, 148, 150, 152, 162, 204  
!SLNFORM, 70  
!SLOW, 74  
!SMODE, 163  
!SMX, 70  
!SORT, 148, 204  
!SPARSE, 61  
!SPATIAL, 71  
!SPECIALCHAR, 35, 41  
!SPLINE, 62  
!SPLIT, 178  
!SQRT link, 88  
!STEP, 62  
!SUB, 48  
!SUBGROUP, 62  
!SUBSECTION, 113  
!SUBSET, 62  
!SUM, 58  
!TABFORM, 71  
!TARGET, 43, 44, 48  
!TAU, 63, 152  
!TDIFF, 178  
!TOLERANCE, 74  
!TOTAL, 88, 89, 90  
!TRIM, 150  
!Ts, 117  
!TSV, 56  
!TURNINGPOINTS, 178  
!TWOStageWEIGHTS, 178  
!TWOway, 71  
!TXTFORM, 71  
!UNIFORM, 48  
!UPPER, 148  
!USE, 113, 118, 177  
!V, 48  
!V target, 44  
!VCC, 71  
!VGSECTORS, 71

# Index

---

!VPGROUP, 104  
!VPV, 178  
!VRB, 74  
!WMF, 63  
!WORK residuals, 91  
!WORKSPACE, 190, 193  
!WVR, 74  
!X, 58  
!XLINKR, 148  
!XML, 190  
!Y, 58  
!YHTFORM, 71  
!YSS, 66, 72  
!YVAR, 190  
PVAL, 62

## R

RAM reduced animal model, 297  
random

correlated effects, 14  
effects, 4, 75, 96

random terms, 81  
multivariate, 143

### RCB

analysis, 25, 105  
design, 22  
model, 28

reading the data, 27, 38

reduced animal model, 297

relationships

genetic, 145  
variance structure parameters, 121

REML, 10, 14

REMLRT, 14

repeated measures, 1, 268

reserved terms, 77

@(*f*), 83  
@(*f,m,n*), 84  
@(*f,n*), 83  
*a(t,r)*, 83  
*abs(v)*, 78, 83  
*and(t,r)*, 83  
*and(t[,r])*, 78  
*at(f)*, 83  
*at(f,m,n)*, 84  
*at(f,n)*, 78, 83  
*c(f)*, 78, 84  
*con(f)*, 84  
*cos(v,r)*, 78, 84  
*fac(v)*, 78, 84  
*fac(v,y)*, 78, 84  
*ge(f)*, 78  
*gt(f)*, 78  
*h()*, 78  
*h(f)*, 84  
*hs(f,k)*, 78  
*i(f)*, 84  
*ide(f)*, 84  
*inv(v[,r])*, 78, 84  
*l(f)*, 84  
*le(f)*, 78  
*leg(v,[-]n)*, 78, 84  
*lin(f)*, 78, 84  
*log(v[,r])*, 79, 84  
*lt(f)*, 79  
*ma1*, 79, 85

*mbf(f)*, 85  
*mbf(f,c)*, 85  
*mbf(v,r)*, 79  
*mu*, 77, 85  
*mv*, 77, 85  
*out(n)*, 79, 85  
*out(n,t)*, 79, 85  
*p(v,[-]n)*, 85  
*p[,o]*, 79  
*pol(v,[-]n)*, 79, 85  
*pow(x,p[,o])*, 79, 86  
*qtl(f,p)*, 79  
*qtl(f,r)*, 86  
*ref(f,k)*, 79  
*s(v[,k])*, 86  
*sin(v,r)*, 79, 86  
*spl(v[,k])*, 78, 86  
*sqrt(v[,r])*, 79, 86  
*t(n)*, 78  
*Trait*, 77, 86  
*uni(f)*, 79  
*uni(f,k[,n])*, 86  
*uni(f[,0[,n]])*, 86  
*units*, 77, 86  
*vect(v)*, 79, 87  
*zero(k)*, 79

reserved words

AEXP, 138  
AGAU, 138  
ANTE1, 139  
ANTE*k*, 139  
AR1, 135  
AR2, 135  
AR3, 135  
ARMA, 136  
CHOL1, 139  
CIR, 138  
CORB, 136  
CORG, 137  
CORGH, 137  
CORU, 136  
DIAG, 138  
EXP, 137  
FA1, 139  
FACV[1], 139  
FACV*k*, 139  
FA*k*, 139  
GAU, 137  
GIV1, 140  
GIV*k*, 140  
GRM1, 140  
GRM*k*, 140  
half, 77  
ID, 135  
IEUC, 138  
IEXP, 137  
IGAUC, 137  
LVR, 138  
MA1, 136  
MA2, 136  
MAT*k*, 138  
NRM, 140  
OWN*k*, 139  
RR1, 139  
RRD*k*, 134  
RRD*k*, 140  
RR*k*, 134, 140  
SAR, 135

# Index

---

- SAR2, 135
- SPH, 138
- US, 139
- XFA1, 139
- XFAk, 139
- residual, 28
  - error, 4, 75, 96
  - likelihood, 11
  - terms, 81
- response variable, 76
- running the job, 29

## S

- sat(), 103
- score
  - test, 58
  - vector, 12, 74
- segmentation fault, 224
- separable
  - autoregressive spatial structure, 9, 102, 107, 217
  - G structure, 108
  - R structure, 9
  - variance structure, 8, 97
- singularities, 93
- sire model, 144
- slow processes, 201
- sparse
  - fixed, 76
  - vs dense, 93
- spatial
  - analysis, 24, 275, 282
  - data, i, 16, 24, 275
  - model, 102, 105, 108, 276
- specifying the data, 38
- split plot design, 259

## T

- tabulation, 27, 171
  - qualifiers, 171
  - syntax, 171
- template file, 25
- timing information, 199
- timing process
  - slow processes, 201

- title line, 26, 38
- TPREDICT, 186
- transformation, 42
  - qualifiers, 44
  - syntax, 44

## U

- unbalanced
  - data, 266
  - nested design, 262
- UNIX, 187
  - crashing, 224
  - debuggin, 191
- unreplicated trial, 282

## V

- variance components
  - calculating functions, 205
  - functions, 205
- variance models
  - combining, 10
  - descriptions, 128
  - forming from correlation models, 128
- variance parameters, 10
  - relationships, 121
- variance structures, 28
  - Matérn, 130, 138
  - multivariate, 142
  - replicate effects, 28
  - residual, 142
  - residual effects, 28
- VCM process, 59, 121

## W

- Wald F statistics, 17, 94
- weight variables, 34
- weighted analyses, 87
- working folder, 53
- workspace options, 193

## X

- XFA extension, 132