VSNi

ASReml for Python

Reference Manual
Version 1.0.0

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

VSN International Ltd

July 29, 2026

ASReml for Python Reference Manual Version 1.0.0

ASReml for Python is developed by VSN International Ltd

Published by

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

email: asreml@vsni.co.uk

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

Copyright notice

Bibliographical reference

The correct bibliographical reference for this document is:

VSN International 2026. ASReml for Python Reference Manual Version 1.0.0. VSN International Ltd., Hemel Hempstead, HP2 4TP, UK.

Companion documents

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

Acknowledgements

ASReml is the culmination of many years of research and software development. We gratefully acknowledge the pioneering work of Arthur Gilmour, Brian Cullis, and Robin Thompson, together with the support of their respective institutions, including NSW Agriculture (now the NSW Department of Primary Industries), the University of Wollongong, and Rothamsted Research. Their vision, expertise, and dedication led to the development of ASReml and established it as one of the leading software packages for fitting linear mixed models using residual maximum likelihood (REML).

Their contributions to the statistical theory, computational algorithms, and practical implementation of mixed models have had a profound impact on quantitative genetics, plant and animal breeding, agriculture, and many other areas of applied statistics. The methods embodied in ASReml continue to influence both research and practice throughout the world.

We also acknowledge the contribution of Dave Butler, Sue Welham and Bev Gogel to ASReml software development and documentation.

The ASReml for Python interface is the result of the dedication and expertise of the team at VSN International. We gratefully acknowledge Darren Murray, Martyn Byng, Giovanni Galli, Salvador Gezan, Vander Fillipe de Souza and David Baird for their efforts in designing, developing, testing, and documenting the Python interface, making the capabilities of ASReml accessible to the wider Python community.

Contents

Preface

ASReml for Python is a statistical package that fits linear mixed models using Residual Maximum Likelihood (REML) in the Python environment. This package uses the same computational kernel as its companion package ASReml-SA. This computational kernel has been under development since 1993 and arose out of collaboration between Arthur Gilmour and Brian Cullis (NSW Department of Primary Industries) and Robin Thompson and Sue Welham (Rothamsted Research) to conduct research into the analysis of mixed models and to develop appropriate software, building on their wide expertise in relevant areas including the development of methods that are both statistically and computationally efficient, the analysis of animal and plant breeding data, the analysis of spatial and longitudinal data and the production of widely used statistical software.

This reference manual documents the features of the methods of class `ASReml`. Outside of the worked examples, it does not consider the statistical issues involved in fitting models.

The features of ASReml for Python include:

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

This manual consists of seven chapters. Chapter 1 introduces ASReml, describes the conventions used throughout the manual, and describes the various data sets used for illustration; Chapter 2 presents a general overview of basic theory; Chapter 3 presents an introduction to fitting models in ASReml followed by a more detailed description of fitting the linear mixed model; Chapter 4 is a key chapter that presents the syntax for specifying variance models for random effects in the model; Chapter 5 describes special functions and methods for genetic analyses; Chapter 6 outlines the prediction of linear functions of fixed and random effects in the linear mixed model; and Chapter 7 presents a comprehensive and diverse set of worked examples. In addition, the Appendix provides additional details on some of the options and functions that can help with specific analyses.

Chapter 1

Introduction

1.1 What ASReml for Python can do

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

- (Un)balanced longitudinal data.
- Repeated measures data (including multivariate analysis and trajectory-type models).
- (Un)balanced designed experiments.
- Multi-environment trials and meta analysis.
- Regular or irregular spatial data.

The computational engine of ASReml for Python is the algorithm of Gilmour et al. (1995) adapted from the standalone program ASReml-SA (Gilmour et al. 2026). The computational efficiency of ASReml arises from using the Average Information REML algorithm (giving quadratic convergence) and sparse matrix operations.

The `asreml` function returns an instance of the class `ASReml`. Results can be obtained using the methods: `residuals`, `fitted`, `coefficients`, `summary`, `variance_components`, `wald` and `predict`.

1.2 Getting started

1.2.1 Installation

Installation instructions are provided on the VSN International website https://vsni.co.uk/support/asreml-python-kb/

1.2.2 Help and references

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

1.2.3 Using this manual

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

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

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

Chapter 2

Some theory

2.1 The general linear mixed model

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

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

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

2.1.1 Sigma parameterization of the linear mixed model

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

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

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

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

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

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

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

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

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

2.1.2 Partitioning the fixed and random model terms

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

2.1.3 G structure for the random model terms

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

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

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

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

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

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

and

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

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

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

2.1.4 Partitioning the residual error term

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

2.1.5 R structure for the residual error term

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

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

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

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

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

2.1.6 Gamma parameterization for the linear mixed model

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

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

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

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

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

In this case

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

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

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

2.1.7 Parameter types

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

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

2.1.8 Variance structures for the random model terms

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

2.1.9 Variance models for terms with several factors

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

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

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

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

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

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

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

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

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

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

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

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

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

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

2.1.10 Direct product structures

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

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

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

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

2.1.11 Direct products in R structures

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

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

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

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

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

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

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

2.1.12 Direct products in G structures

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

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

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

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

2.1.13 Range of variance models for R and G structures

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

Among the range of correlation models are:

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

Among the range of covariance models are:

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

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

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

2.2 Estimation

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

2.2.1 Estimation of the variance parameters

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

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

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

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

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

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

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

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

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

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

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

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

where

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

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

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

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

We can also write

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

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

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

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

The elements of the observed information matrix are

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

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

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

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

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

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

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

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

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

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

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

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

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

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

2.2.2 Estimation/prediction of the fixed and random effects

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

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

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

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

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

These can be written as

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

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

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

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

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

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

2.3 What are BLUPs?

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

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

where $\bar{\boldsymbol{y}}$ is the vector of treatment means and $\bar{y}_{..}$ is the grand mean. The differences of the treatment means and the grand mean are the estimates of treatment effects if treatment effects are fixed. The BLUP is therefore a weighted mean of the data based estimate and the *prior* mean $\boldsymbol{A}\boldsymbol{\psi}$. If $\boldsymbol{\psi} = \mathbf{0}$, the BLUP in (2.18) becomes

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

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

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

2.4 Inference: fixed effects

2.4.1 Introduction

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

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

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

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

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

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

2.4.2 Incremental and conditional Wald statistics

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

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

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

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

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

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

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

and

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

Implicit in these calculations is that

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

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

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

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

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

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

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

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

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

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

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

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

Consider now a nested model which might be represented symbolically by

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

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

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

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

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

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

2.4.3 Kenward and Roger adjustments

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

2.5 Inference: random effects

2.5.1 Tests of hypotheses: variance parameters

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

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

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

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

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

which is strictly positive. If r_i is the number of parameters estimated in model i $(i = 1, 2)$, then the asymptotic distribution of the REMLRT, under the restricted model is $\chi^2_{r_2 - r_1}$.

The REMLRT is implicitly two-sided, and must be adjusted when the test involves a hypothesis with the parameter on the boundary of the parameter space. It can be shown that for a single

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

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

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

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

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

2.5.2 Diagnostics

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

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

It follows that

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

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

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

2.5.3 Variogram

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

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

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

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

ASReml also computes the variogram from predictors of random effects which appear to have a variance structure defined in terms of distance. The coordinates can be accessed using the `variogram()` function (see Section 7.5).

Chapter 3

Fitting the mixed model

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

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

3.1 The data frame

Data for analysis using ASReml are generally contained in a text file or a spreadsheet and are read into a *data frame* using the appropriate Pandas functions.

Six rows of the NIN field trial data (available via `nin89`) are reproduced below. Note that the data are in field order (rows within columns) and a header line (first row) is included. In this case there are 12 comma-separated data fields (`Plot`...`Column`) and the complete file has 242 data rows, one for each variety in each replicate.

	Plot	Variety	Id	pid	raw	Rep	nloc	yield	lat	long	Row	Column
15	16	LANCER	1	1101.0	585.0	1.0	4	29.25	4.3	19.2	16	1
16	17	BRULE	2	1102.0	631.0	1.0	4	31.55	4.3	20.4	17	1
17	18	REDLAND	3	1103.0	701.0	1.0	4	35.05	4.3	21.6	18	1
18	19	CODY	4	1104.0	602.0	1.0	4	30.10	4.3	22.8	19	1
19	20	ARAPAHOE	5	1105.0	661.0	1.0	4	33.05	4.3	24.0	20	1
20	21	NE83404	6	1106.0	605.0	1.0	4	30.25	4.3	25.2	21	1

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

A data frame is internally created from a text file source using a Pandas function call like:

```
import pandas as pd

nin89 = pd.read_csv("/path/nin89.csv")
```

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

- Blank lines are ignored.
- It is sensible to include a header line in the data file; if no header line is included use the `header=None` argument and the columns will be labelled as `0, 1, 2, ..., n-1` where n is the number of columns.
- The same column name should not be repeated.
- `NaN` is the default code for missing values.
- In comma-separated text (.csv) files:
  - Consecutive commas imply a missing value.
  - Provided the number of fields is consistent, a line beginning (ending) with a comma will generate `NaN` for that observation in the first (last) variate or a zero length string if a text field.
- Blanks may be embedded in text fields provided the field delimiter is not also the space character, otherwise the string must be enclosed in quotes.
- Too many or too few data fields on a line will cause an error.

3.1.1 Covariates

For analysis purposes it is recommended that covariates be centred and rescaled to have a mean of 0 and a variance of 1 to avoid failure by potential singularities. In addition, missing values in covariates are replaced with zeros so it is important in these circumstances to centre the covariate in question. For example, the command

```
nin89['linrow2'] = nin89['Row'].astype(float) - nin89['Row'].mean()
```

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

3.1.2 Categorical variables or factors

Some variables need to be identified as categorical variables or factors for the model fitting. To define which columns of the data set will be transformed into factors, we use the `define_categorical_variables()` function, as follows:

```
from asreml import define_categorical_variables

nin89 = define_categorical_variables(nin89,
              variables=["Plot", "Variety", "Id", "Rep", "Row", "Column"])
```

which creates a dictionary with two items holding the categorized data frame and category labels. The item 'data' contains the pandas data frame with the variables `Plot`, `Variety`, `Id`, `Rep`, `Row` and `Column` recoded as categories. The item 'categories' is a dictionary holding the associated factor level/labels.

3.2 Introducing the `asreml()` function call

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

```
import asreml

nin89_asr = asreml(
  response="yield",
  fixed="Variety",
  random="idv(Rep)",
  residual="idv(units)",
  options={"na_action": {"x": "include"}},
  data=nin89
)
```

where `nin89_asr` is the name we have chosen for the returned object, and `idv()` is the homogeneous identity variance structure (assumed by default).

The linear model is specified in the `response` (required), `fixed` (optional), `random` (optional) and `residual` (optional error component) arguments. A fourth optional model argument `sparse` is also available but is not used explicitly (see also Section 3.4.2) in this example.

The name of the response variable is supplied as a string using the argument `response`. The `fixed`, `random`, `residual` and `sparse` arguments require a formula string where the names of terms are separated by `+` operators. In this case the response name is `yield`, and `Variety` is a fixed term in the model, so the `response` and `fixed` arguments are given by:

```
nin89_asr = asreml(
  response="yield",
  fixed="Variety",
  ...
)
```

The random terms in the model are specified as a formula string. In this example `Rep` is a random term and the `random` argument is

```
nin89_asr = asreml(
  ...,
  random="idv(Rep)",
  ...
)
```

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

```
nin89_asr = asreml(
  ...,
  residual="idv(units)",
  ...
)
```

3.2.1 Methods

The following methods are available to extract results: `.summary()`, `.trace()`, `.coefficients()`, `.fitted()`, `.residuals()`, `.variance_components()`, `.wald()`, `.vpredict()`, and `.predict()` (see Chapter 6).

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

```
summary_nin89 = nin89_asr.summary()
summary_nin89["varcomp"]
```

or

```
nin89_asr.variance_components()
```

3.3 A note on data order

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

```
asreml(
  ...
  residual="ar1v(Column):ar1(Row)",
  ...
)
```

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

3.4 Fixed terms

3.4.1 Dense fixed terms

The fixed model formula specifies the fixed factors, interactions and covariates for which standard errors and tests of significance are required. The fixed model formula must contain at least one term which may simply be the intercept. By default the intercept is included in the fixed model; for example,

```
asreml(
  ...
  fixed="Variety",
  ...
)
```

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

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

An intercept-only fixed model is specified by including only a `1` or 'mu' , for example,

```
asreml(
  ...
  fixed="1",
  ...
)
```

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

```
asreml(
  ...
  fixed="lin(Row)",
  ...
)
```

Model functions also exist to generate orthogonal polynomials (`pol()` and `leg()`) or splines (`spl()`) and to fit terms conditionally (`at()`; see Section 3.6).

3.4.2 Sparse fixed terms

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

3.5 Random terms

The `random` model formula specifies the names of the factors, interactions, covariates and special terms that comprise the random component of the model. All of these effects are estimated using sparse matrix methods. Each random term will have a variance model associated with it which, when no variance model function is specified, defaults to a scaled identity $\gamma \boldsymbol{I}_n$ or $\sigma^2 \boldsymbol{I}_n$ where γ is a variance ratio, depending on whether a sigma or gamma parameterization is used for fitting. This is equivalent to `idv()` and further details are presented in the types of variance models in Section 4.2.

3.5.1 Initial values and constraints for variance parameters

Initial values and constraints for variance parameters are generated automatically and they depend on the parameterization used. For example, the default initial values are 0.1 for variance ratios (when

parameters are estimated using the gamma parameterization) and $0.1 * \mathsf{v}$ for variance components (when parameters are estimated using the sigma parameterization), where v is half the raw variance of the response. Using both parameterizations correlations are assumed to have a starting value of 0.1. The corresponding default parameter constraints are P (positive) for variance component ratios, U (unconstrained) for correlations and P for variance components.

The example below is for a RCB field trial analysis

```
nin89_asr = asreml(
  response="yield",
  fixed="Variety",
  random="idv(Rep)",
  residual="units",
  options={"na_action": {"x": "include"}},
  data=nin89
)
```

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

3.5.2 Replacing elements in an internal object

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

```
nin89_asr = asreml(
  response="yield",
  fixed="Variety",
  random="idv(Rep)",
  residual="units",
  options={"na_action": {"x": "include"}},
  data=nin89,
  fit_options={"process_stage": "preprocess"}
)
nin89_ini = nin89_asr.initial_values(display=True)
```

returns a dictionary holding initial value information, each key in the dictionary is a variance component label and each value is a dictionary with one or both keys “value” and “constraint” holding the initial value and the constraint code. Constraint codes are: “P” - in the space, attempts to keep the parameter in the theoretical parameter space; “U” - unrestricted and “F” - fixed.

Elements of the initial values dictionary can be modified to set new values and then the `fit` method can be used to run the analysis. The example below shows how the initial value for `Rep` can be changed and the model run using the `fit` method.

```
nin89_ini["idv(Rep)!IDV_V"]["value"] = 0.5
nin89_asr.fit(initial_values=nin89_ini)
```

3.5.3 Specifying variance structures

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

```
asreml(
  ...
  residual="ar1v(Column):ar1(Row)",
  ...
)
```

(assuming the data are correctly ordered as `Row` within `Column`) in the call, where `ar1v()` and `ar1()` are part of a special function specifying a first order autoregressive variance model for both `Column` and `Row` (see Section 4.1). The complete range of possible variance models is presented in Table B.1.

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

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

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

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

It is conventional to use a variable in the data file to identify sections within the data. The data will be sorted internally by ASReml (*i.e.*, the data file does not need to be ordered in any particular

way) and the variance structures for sections can then be specified using the `dsum()` function, for example:

```
asreml(
  ...
  residual="dsum(units, section)",
  ...
)
```

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

- It tells ASReml that the variance structure for the residual error term is a direct sum structure where different variance structures apply to the different levels of the sectioning variable in the data.
- If a model structure specified defines a residual matrix then a variance factor associated with the appropriate sectioning level is added to the specified model to generate a variance matrix.
- It prunes the levels for a section so that *only* the levels of factors defining the residual variance structure for that section are used in forming that variance structure.

3.6.1 Specifying the model using `dsum`

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

```
residual="dsum(ar1v(Column):ar1(Row), site)"
```

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

```
residual="dsum(ar1(Column):ar1(Row), site)"
```

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

```
residual="dsum(ar1(Column):ar1(Row) + id(Column):ar1(Row), Site, levels = [[1,
3],[2]])"
```

or

```
residual="dsum(ar1(Column):ar1(Row) + id(Column):ar1(Row), Site, levels =
[['Site1', 'Site3'],['Site2']])"
```

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

```
residual="dsum(ar1(Column):ar1(Row), Site, levels = [1, 3]) +
dsum(id(Column):ar1(Row), site, levels = [2])"
```

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

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

```
residual="dsum(ar1v(Column):ar1(Row), Site, levels = ['Site1', 'Site3'])"
```

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

3.6.2 Providing starting values for `dsum`

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

```
residual="dsum(ar1v(Column, init = [0.9, 12]):ar1(Row, init = 0.9) + idv(Column,
init = 12):ar1(Row, init = 0.5), Site, levels = [[1, 3], [2]])"
```

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

3.7 Two rules for defining the residual error term

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

- Rule 1 The number of effects in the residual term *must* be equal to the number of data units included in the analysis.

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

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

3.8 Formula operators in fixed, random, residual and sparse arguments

The combination of factors and covariates in the fixed, random, residual and sparse model arguments can be compacted using formula for combining, removing, interacting, crossing and nesting along with expansion of bracketed terms.

- `:` The interaction of two terms, A:B gives groups for each of the unique combinations of A and B.
- `+` The addition of two terms, A+B gives separate effects for A and B.
- `-` The removal of a term, -B removes effects for B.
- `*` The crossing of two terms, A*B gives the factorial combinations of A and B = A + B + A:B.
- `/` The nesting of two terms, A/B is equivalent to A + A:B (the main effect of A with differences for B within A).
- `()` The bracketed expansion to condense terms, for example A:(B+C) = A:B + A:C and (A+B)*(C+D) = A*C + A*D + B*C + B*D.

These symbols can be combined to create complex combinations of terms as shown in the following examples:

- A*B*C = A + B + C + A:B + A:C + B:C + A:B:C
- A/B/C = A + A:B + A:B:C
- A*B/C = A + B + A.B + A:C + B:C + A:B:C

- (A/B):(C/D) = A:C + A:C:D + A:B:C + A:B:C:D
- (A/B)*(C/D) = A + A:B + C + C:D + A:C + A:B:C + A:C:D + A:B:C:D
- A*(B + C*D) = A + B + C + D + A:B + A:C + A:D + C:D + A:C:D
- -(A*B) = -A - B - A:B

3.9 Diagnostic plots

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

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

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

```
nin89_asr = asreml(
  response="yield",
  fixed="Variety",
  random="idv(Rep)",
  residual="idv(units)",
  options={"na_action": {"x": "include"}},
  data=nin89
)

residual_plot(nin89_asr)
```

A composite plot of the diagnostic output described above can be obtained with `residual_plot(nin89_asr)`. Here, we can see a few observations to the left that appear to be outside of the distribution. Further exploration is required to determine the reason. To explore the individual residual values it is possible to use `.residuals()` with an `asreml` object.

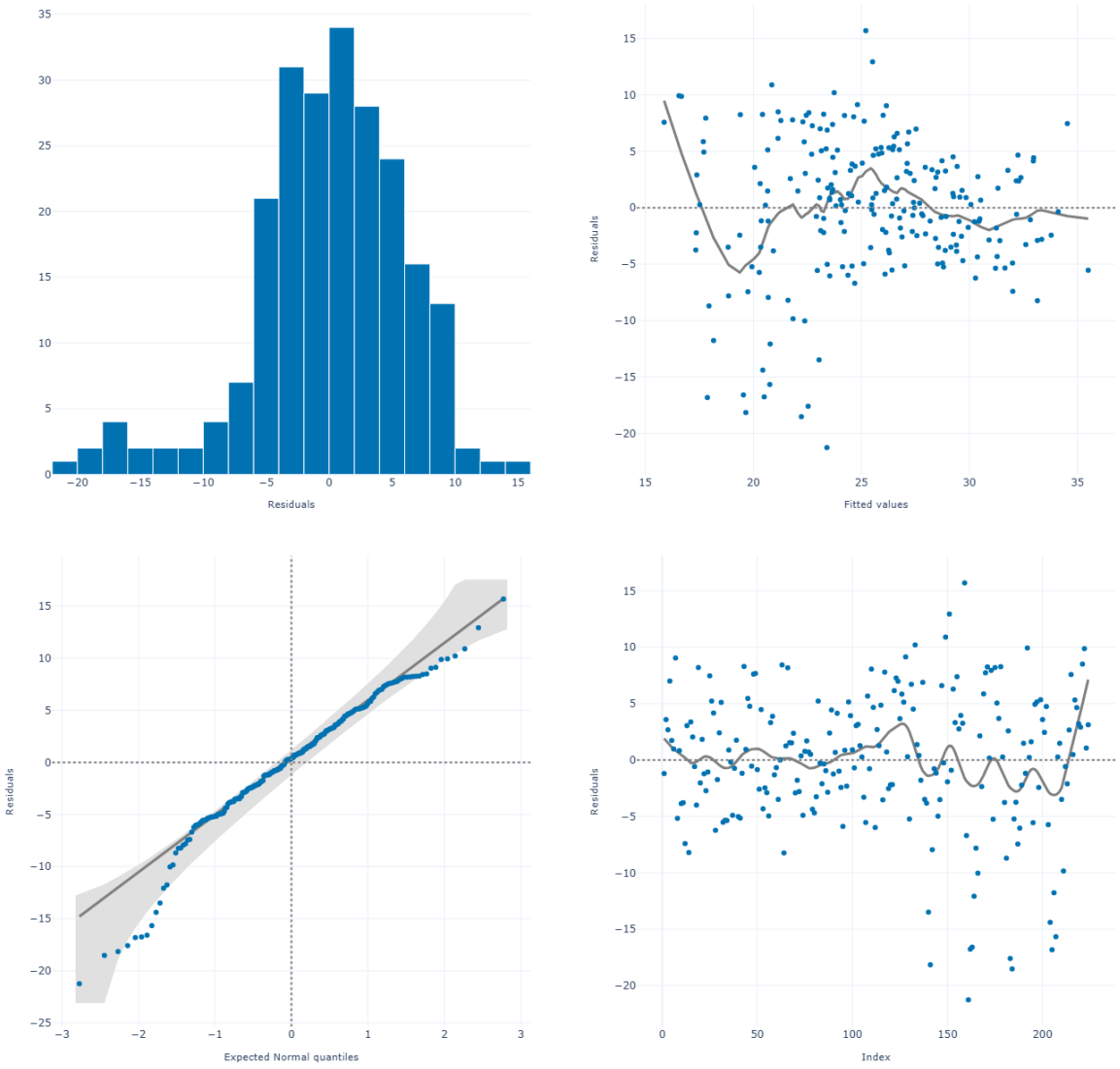

Figure 3.1: Residual Plots

The method `residual_plot` presents, by default, the raw or `working` residuals, but there are other options within `asreml`, further details can be found in the Python help. If the residual structure of the model contains multiple sections, the default plots are conditioned on the factor whose levels define the sections.

3.10 Weights

Weighted analyses are achieved by using the `weights` argument to `asreml()`. If these are relative weights (to be scaled by the units variance) then this is all that is required; for example, the number of sampling units: `wt = [3, 1, 3,...]`. If they are absolute weights, that is, the reciprocal of known variances, the units variance should be constrained to 1.

```
asreml(
  ...,
  weights="wt",
  ...
)
```

3.11 Missing values

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

Missing values in the response

Records with missing values in the response (`y`) are omitted by default, `options={"na_action": {"y": "omit"}}`, which excludes units with missing values in the response. An alternative action is `options = {"na_action": {"y": "include"}}`, which retains these records and estimates them as a consequence of fitting the model: a factor labelled `mv` is created and included in the sparse equations. Note that missing values must be estimated in a multivariate analysis, so the default "omit" option will produce some errors there and "include" should be used.

Missing values in the explanatory variables

Covariates - Records with missing values in covariates `x` are only discarded if `options={"na_action": {"x": "omit"}}`. If included, they are treated as zeros which may only be reasonable if the covariate values are centred.

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

3.12 Generalized linear (mixed) models

3.12.1 GLMs

The main function `asreml()` includes the ability to use family functions for fitting Generalized Linear Models (GLM) (McCullagh and Nelder 1994). The family distributions that can be used are shown in Table 3.1.

Table 3.1: Family distributions and functions

Distribution	Function
Normal	gaussian(link = "identity", dispersion = NaN)
Gamma	gamma(link = "inverse", dispersion = 1, phi = 1)
Binomial	binomial(link = "logit", dispersion = 1, total = None)
Negative binomial	negative_binomial(link = "log", dispersion = 1, phi = 1)
Poisson	poisson(link = "log", dispersion = 1)

Table 3.2 lists the link functions that can be used to connect the linear predictor $\boldsymbol{\eta}$ to the mean ($\boldsymbol{\mu}$) on the data scale.

Table 3.2: Families and link functions

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

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

3.12.2 GLMMs

There is the capacity to fit a wider class of models which include additional random effects for non-Normal error distributions. The inclusion of random terms in a GLM is usually referred to as a Generalized Linear Mixed Model (GLMM). For GLMMs, `asreml()` uses what is commonly referred to as penalized quasi-likelihood or PQL (Breslow and Clayton 1993). The argument `family` is used to define the distribution considered for the GLM or GLMM models. For example, the data frame `binnor` has a binary response named `score4` that corresponds to the incidence of footshape class 1. The corresponding GLMM model for this data is:

```
binnor_asr = asreml(
  response="score4",
  fixed="Sex + Grp",
  random="Sire",
  family="binomial(link = logit, dispersion = 1)",
  data=binnor
)
```

The `asreml()` function has implemented the technique penalized quasi-likelihood or PQL (Breslow and Clayton 1993). This technique can give somewhat misleading results in certain situations, therefore we recommend that GLMMs be used only by advanced users, and with care.

3.13 Multivariate analyses

Multivariate analyses are used when we are interested in estimating the correlations between distinct traits (for example, fleece weight and fibre diameter in sheep) and for repeated measures of a single trait. The term *multivariate* analysis is used here in the narrow sense of a multivariate mixed model.

3.13.1 Model specification

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

```
asreml(
  response=["y1",...,"yk"],
  ...
)
```

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

3.13.2 A bivariate example

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

```
wether_asr = asreml(
  response=["gfw", "fdiam"],
  fixed="trait + trait:Year",
  random="us(trait,init = [0.4,0.3,1.3]):Team + us(trait,init =
[0.2,0.2,2.0]):Tag",
  residual="id(units):us(trait,init = [0.2,0.2,0.4])",
  predict={"classify" : "trait:Year"},
  data=owt
)
```

3.13.3 A repeated measures example

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

```
wolfinger_asr = asreml(
  response=["wt0", "wt1", "wt2", "wt3", "wt4"],
  fixed="trait + Treatment + trait:Treatment",
  residual="id(units):us(trait)",
  data=wolfinger
)
```

3.13.4 Specifying multivariate variance structures

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

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

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

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

- The error structure must be specified as two-dimensional, with independent units and often an unstructured variance matrix across traits.
  - The `residual` for this model is therefore `residual="id(units):us(trait)"`.
  - Missing values are allowed and *must* be fitted; `asreml()` automatically includes the special factor `mv` in the `sparse` formula in such cases.

- For the default analysis where the response is specified as a matrix, the R structure *must* reflect the data order of *traits within units* which means that the term `units` must appear before `trait` in the `residual` formula.
- Variance parameters are variances, not variance ratios.
- The error structure is often specified as an unstructured variance matrix but correlation models may also be used. `asreml()` attempts to detect such cases and fix or estimate the residual scale parameter accordingly.

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

```
wolfinger_asr = asreml(
  response=["wt0", "wt1", "wt2", "wt3", "wt4"],
  fixed="trait + Treatment + trait:Treatment",
  residual="id(units):ar1v(trait)",
  data=wolfinger
)
```

- Initial values for the variance matrices are given as the lower triangle of the (symmetric) matrix specified row-wise.
- Nominating reasonable initial values can be a problem. By default, `asreml()` uses half of the phenotypic variance-covariance matrix in forming initial values.

3.14 Testing of terms: the `wald` method

```
asreml(
  ...
  options={"wald": {"den_df": "numeric", "ss_type": "conditional"}},
  ...
)
```

The results returned by the `wald` method depend on the options specified for the `den_df` and `ss_type` arguments.

Incremental F-statistics

If `den_df = None` and `ss_type = "incremental"`, a table of Wald statistics for fixed effect terms is returned. These are tested sequentially, which means that factors are adjusted for terms higher in the table (or not in the table), but ignoring terms that occur below.

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

Conditional F-statistics

If a different combination of `den_df` and `ss_type` is used then the results returned will include columns for the approximate denominator degrees of freedom or conditional F-statistics depending on the combination of options chosen.

The results have four styles:

```
Source of Variation  NumDF          F-inc
Source of Variation  NumDF          F-inc          F-con  M
Source of Variation  NumDF   den_df          F-inc  P-inc
Source of Variation  NumDF   den_df          F-inc          F-con  M  P-con
```

depending on whether conditional F-statistics are reported or whether the denominator degrees of freedom are calculated. See Section 2.4 for more background on the contents of this table.

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

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

```
wether_asr = asreml(
  ...
  options={"wald": {"den_df": "numeric", "ss_type": "conditional"}},
  ...
)

wether_asr.wald()
```

Chapter 4

Specifying variance structures

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

- The mixed linear model
$$\boldsymbol{y} = \boldsymbol{X}\boldsymbol{\tau} + \boldsymbol{Z}\boldsymbol{u} + \boldsymbol{e}$$
has a residual term
$$\boldsymbol{e} \sim N(\mathbf{0},\ \boldsymbol{R}_v(\boldsymbol{\sigma}_r))$$
and random effects
$$\boldsymbol{u} \sim N(\mathbf{0},\ \boldsymbol{G}(\boldsymbol{\sigma}_g))$$
where, in the most complex forms
$$\boldsymbol{R}_v = \oplus_i \boldsymbol{R}_{v_i}$$
$$\boldsymbol{G} = \oplus_j \boldsymbol{G}_j$$
and each
$$\boldsymbol{R}_{v_i} = \boldsymbol{R}_{v_i}(\boldsymbol{\sigma}_{r_i})$$
$$\boldsymbol{G}_j = \boldsymbol{G}_j(\boldsymbol{\sigma}_{g_j})$$
where $\boldsymbol{\sigma}_{r_i}$ and $\boldsymbol{\sigma}_{g_j}$ parameterize the respective variance models for each section.
- We use the terms *R structure* and *G structure* to refer to the matrices $\boldsymbol{R}_{v_i}$ and $\boldsymbol{G}_j$ above in a syntactic manner, respectively.
- R and G structures are typically formed as a direct product of particular variance models.
- The order of terms in a direct product must agree with the order of factors in the corresponding model term.
- Variance models may be correlation matrices, or variance matrices with equal or unequal variances on the diagonal. A model for a correlation matrix (e.g. `ar1()`) can be converted to an equal (homogeneous) variance form (*e.g.* `ar1v()`) and to a heterogeneous variance form (*e.g.* `ar1h()`).

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

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

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

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

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

4.1 A sequence of structures for the NIN field trial data

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

This section illustrates:

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

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

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

```
rcb_asr = asreml(
  response="yield",
  fixed="Variety + Rep",
  residual="idv(units)",
  options={"na_action": {"x": "include"}},
  data=nin89
)
```

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

Model 1b: RCB analysis with G and R structures

```
rcb_asr = asreml(
  response="yield",
  fixed="Variety",
  random="idv(Rep)",
  residual="idv(units)",
  options={"na_action": {"x": "include"}},
  data=nin89
)
```

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

Note that when specifying G structures ASReml automatically adds a scale parameter if a correlation model is specified (see Section 4.1.1 for more details). Only one of the models specified in a G structure can be a variance model. If the variance matrix of a term contains several component matrices then the problem of *identifiability* arises.

For example, a random term

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

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

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

```
sp_asr = asreml(
  response="yield",
  fixed="Variety",
  residual="idv(Column):ar1(Row)",
  options={"na_action": {"x": "include","y": "include"}},
  data=nin89
)
```

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

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

Model 2b: two-dimensional spatial model

```
sp_asr = asreml(
  response="yield",
  fixed="Variety",
  residual="ar1v(Column):ar1(Row)",
  options={"na_action": {"x": "include","y": "include"}},
  data=nin89
)
```

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

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

```
sp_asr = asreml(
  response="yield",
  fixed="Variety",
  random="idv(units)",
  residual="ar1v(Column):ar1(Row)",
  options={"na_action": {"x": "include","y": "include"}},
  data=nin89
)
```

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

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

```
sp_asr = asreml(
  response="yield",
  fixed="Variety",
  random="ar1v(Column):ar1(Row)",
  residual="idv(units)",
  data=nin89
)
```

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

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

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

	`asreml` arguments	random term (G)			residual error term (R)		
			model			model	
			1	2		1	2
1a	`fixed = "Variety + Rep"`	-	-	-	units	idv()	-

1b	`fixed = "Variety",`
	`random = "idv(Rep)",`
	`residual = "idv(units)"`
		Rep	idv()	-	units	idv()	-

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

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

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

4.1.1 Reducing the specification using defaults

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

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

```
rcb_asr = asreml(
  response="yield",
  fixed="Variety",
  random="Rep",
  options={"na_action": {"x": "include"}},
  data=nin89
)
```

and

```
rcb_asr = asreml(
  response="yield",
  fixed="Variety",
  random="Rep",
  residual="idv(units)",
  options={"na_action": {"x": "include"}},
  data=nin89
)
```

are equivalent to

```
rcb_asr = asreml(
  response="yield",
  fixed="Variety",
  random="idv(Rep)",
  residual="idv(units)",
  options={"na_action": {"x": "include"}},
  data=nin89
)
```

and

```
rcb_asr = asreml(
  response="yield",
  fixed="Variety",
  random="idv(Rep)",
  residual="units",
  options={"na_action": {"x": "include"}},
  data=nin89
)
```

4.2 Types of variance models

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

4.2.1 Correlation models

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

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

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

4.2.2 Homogeneous variance models

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

That is,

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

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

4.2.3 Heterogeneous variance models

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

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

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

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

4.2.4 Positive definite matrices

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

4.3 Variance model functions

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

4.3.1 Default identity

```
id(variable_name)
idv(variable_name, init = None)
idh(variable_name, init = None)
```

Required arguments

`variable_name` a name of a factor in the primary dataset.

Optional arguments

`init` a vector of initial parameter values.

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

Details

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

4.3.2 Time series type models

```
ar1(variable_name, init = None)
ar2(variable_name, init = None)
ar3(variable_name, init = None)
sar(variable_name, init = None)
sar2(variable_name, init = None)
ma1(variable_name, init = None)
ma2(variable_name, init = None)
arma(variable_name, init = None)
```

Description

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

Required arguments

`variable_name` a name of a factor in the primary dataset.

Optional arguments

`init` a vector of initial parameter values.

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

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

```
exp(x, init = None, dist = None)
gau(x, init = None, dist = None)
lvr(x, init = None, dist = None)
iexp(x, y, init = None)
igau(x, y, init = None)
ieuc(x, y, init = None)
sph(x, y, init = None)
cir(x, y, init = None)
aexp(x, y, init = None)
agau(x, y, init = None)
mtrn(x, y, phi = None, nu = 0.5, delta = 1.0, alpha = 0.0, lambda = 2)
```

Description

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

Required arguments

`x` a continuous variable in the primary dataset containing the x coordinates.

`y` a continuous variable in the primary dataset containing the y coordinates.

Optional arguments

`init` a vector of initial parameter values.

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

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

`phi` the range parameter. Default: $\phi = None$.

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

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

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

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

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

Details

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

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

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

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

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

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

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

4.3.4 General structure models

```
cor(variable_name, init = None)
corb(variable_name, k = 1, init = None)
corg(variable_name, init = None)
diag(variable_name, init = None)
us(variable_name, init = None)
chol(variable_name, k = 1, init = None)
cholc(variable_name, k = 1, init = None)
ante(variable_name, k = 1, init = None)
sfa(variable_name, k = 1, init = None)
fa(variable_name, k = 1, init = None)
facv(variable_name, k = 1, init = None)
rr(variable_name, k = 1, init = None)
```

Description

The class of general variance models includes the simple, banded and general correlation models (`cor`, `corb`, `corg`), the diagonal, unstructured, Cholesky and antedependence variance models (`diag`, `us`, `chol`, `cholc`, `ante`) and the factor analytic structure (`fa`) and its variants (`sfa`, `facv`, `rr`).

Required arguments

`variable_name` a name of a factor in the primary dataset.

Optional arguments

`init` a vector of initial parameter values.

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

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

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

Details

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

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

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

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

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

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

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

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

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

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

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

$$\boldsymbol{\Sigma}^{-1} = \boldsymbol{U}\boldsymbol{D}\boldsymbol{U}'$$

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Limitations

The functions `fa(, k)` and `sfa(, k)`, unlike `facv(, k)`, do not allow singular $\boldsymbol{\Sigma}$ matrices to be estimated. In addition, constraints are required in $\boldsymbol{\Gamma}$ for $k > 1$ for identifiability. These are automatically set unless the user ensures identifiability by constraining one parameter in the second column, two in the third column, etc. Unfortunately because the `fa(, k)` formulation allows singular variance matrices it is not available in residual (R) structures.

Updating loadings in factor analytic models

The algorithm for updating loadings in factor analytic models modifies the average information matrix by increasing the diagonal elements pertaining to loadings by a percentage, p. The default is to start with $p = 10\%$ and reduce it by 1 or 2% each iteration down to 1%. If the starting values are poor, 10% may not be a sufficient initial retardation. If it appears the updates are unreasonable, ASReml will increase the value of p by 10% and then continue.

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

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

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

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

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

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

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

4.3.6 Known relationship structures

`vm(variable_name, source, singG='PD', invert=False, ldet=0.0)`

Required arguments

`variable_name` a name of a categorical variable / factor in the primary dataset.

`source` The known inverse of a relationship matrix in the form of a lower triangular sparse matrix held in three-column coordinate form in row-major order.

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

`invert` If true, then the matrix supplied in source will be inverted before being used.

`ldet` The log determinant of the matrix supplied in source. If not supplied the log determinant will be calculated (this can be numerically intensive).

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

```
nassau = pd.read_csv("nassau-Python.csv")

nassau = define_categorical_variables(nassau,
    variables=["rep","iblk","culture","Clone"])
```

```
nassau_sparse = pd.read_csv("nassau_sparse.csv")
```

```
uid = pd.read_csv("nassau_uid.csv")
uid = uid['CloneID'].to_list()
uid = list(map(str.lower, uid))
```

```
grm = {"inverse": nassau_sparse, "ids": uid}

nassau_asr = asreml(
  response="HT6",
  fixed="mu + culture + culture:rep",
  random="vm(Clone, grm) + idv(Clone) + id(rep):idv(iblk)",
  predict={"classify": "culture:rep"},
  options={"maxiterations": 30, "gdense": True},
  data={"data": nassau, "grm": grm}
)

nassau_asr.summary()
```

4.3.7 Variance structures spanning several model terms

`str(form, vmodel)`

Required arguments Arguments: form: A model formula included verbatim in the asreml() random argument vmodel: A direct product variance model for the set of terms given in form

`form` a model formula included verbatim in the asreml() random argument that collectively will have an associated variance model.

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

Details

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

For example, in the first order random coefficient regression model it is required to specify a covariance between the intercept and slope for each subject to ensure translation invariance, that is,

equivalent variance parameter estimates for addition of any constant to the independent variable. For example, in a random coefficient regression where a set of random intercepts is specified by the model term `Subject` (with 10 levels) and a set of random slopes is specified by the model term `time:Subject` (`time` is a variate). Here, translation invariance is achieved using `str()` as

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

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

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

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

Here, the set of subject intercepts has a common variance (σ_{II}), and the set of subject slopes has a (different) common variance (σ_{SS}). Intercepts and/or slopes from two different subjects are independent, but the intercept and slope from any given subject have covariance σ_{IS} (or correlation $\sigma_{IS}/\sqrt{\sigma_{II}\sigma_{SS}}$). In this context, we use integers as input to emphasize that the arguments are specifying the size of the variance structure. For this example, `id(10)` can be replaced by `id(Animal)`.

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

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

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

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

```
data_df['half'] = 0.5
```

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

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

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

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

4.4 Rules for combining variance models

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

- When combining variance models in both R and G structures, the resulting direct product structure must match the ordered effects with the outer factor first. For example, the NIN data are ordered rows within columns. This is why in **Model 3** (page 51) the `ar1v()` variance model for `Column` is specified first in the interaction term.
- ASReml automatically includes and estimates an error variance parameter for each section of the R structure that is a correlation matrix.
- When the G structure involves more than one variance model, one must be either a homogeneous or a heterogeneous variance model and the rest should be correlation models; if more than one are non-correlation models then constraints should be used to avoid identifiability problems, that is, to prevent attempts to estimate confounded parameters.

4.5 Default initial values and constraints for variance parameters

The default initial values are 0.1 for both variance ratios and correlations, and $0.1 * v$ for variance components, where v is half the variance of the response. The corresponding default parameter constraints are `P` (positive) for variance ratios, `U` (unconstrained) for correlations and `P` for variance components. These defaults can be altered using the methods described in Section 3.5.1 and particularly for complex or unbalanced analyses it is recommended that starting values are provided to facilitate convergence.

4.5.1 Providing initial values for variance structures

In ASReml, the majority of the variance models accept some starting values. Providing good starting values facilitates the convergence of the mixed model, and for large and complex models, the model

fit might finish more quickly. Therefore, we recommend its use in most cases, particularly with unbalanced data. Starting values can be difficult to determine, but one strategy is to fit simpler models and use these.

For `asreml()`, a single value or a vector of values can be provided, and these are assigned to the argument `init`. For more than one value it is important to match exactly the order in which the parameters are defined.

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

Example of declaring the initial value inside idv():

```
rcb_asr = asreml(
  response="yield",
  fixed="Variety",
  random="idv(Rep, init = 10)",
  residual="units",
  options={"na_action": {"x": "include", "y": "include"}},
  data=nin89
)
```

Example of declaring a list of initial values within the data dictionary:

```
initials = [0.90, 60, 73, 308, 434, 380]
grass_mod = asreml(
  response="y",
  fixed="Tmt + Time + Tmt:Time",
  residual="id(Plant):exph(Time, my_initials)",
  data={"data": grassUV, "my_initials": initials}
)
```

Another possibility is to use `fit_options="process_stage": "preprocess"` and `grass_ini = grass_mod.initial_values(display = True)` to get a table with the variance structure values and after setting the initial values, run the model again using `grass_mod.fit(initial_values = grass_ini)`.

```
grass_mod = asreml(
  response="y",
  fixed="Tmt + Time + Tmt:Time",
  residual="id(Plant):exph(Time)",
  data=grassUV,
  fit_options={"process_stage": "preprocess"}
)
```

```
grass_ini = grass_mod.initial_values(display=True)
```

```
grass_ini["id(Plant):exph(Time)!Residual!SCA_V"]["value"] = 60
grass_ini["id(Plant):exph(Time)!Time_1!EXP_R"]["value"] = 0.90
grass_ini["id(Plant):exph(Time)!Time_1!EXP_V"]["value"] = 60
grass_ini["id(Plant):exph(Time)!Time_2!EXP_V"]["value"] = 73
grass_ini["id(Plant):exph(Time)!Time_3!EXP_V"]["value"] = 308
grass_ini["id(Plant):exph(Time)!Time_4!EXP_V"]["value"] = 434
grass_ini["id(Plant):exph(Time)!Time_5!EXP_V"]["value"] = 380
grass_mod.fit(initial_values=grass_ini)
```

4.5.2 The vcm argument to `asreml()`

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

```
tulli_mod = asreml(
    response="yield",
    fixed="mu + Covariate + mv",
    random="Check + repchk + units",
    residual="ar1(Column, 0.8315):Row",
    options={"na_action": {"x": "include"}, "maxiterations": 8},
    fit_options={
        "initial_values": {
            "Check!IDV_V": {
                "value": 1.0,
                "vcm": [1]},
            "repchk!IDV_V": {
                "value": 1.0,
                "vcm": [-1]},
            "units!IDV_V": {
                "value": 1.0},
            "ar1(Column):Row!Column!AR_R": {
                "value": 0.8315}
        }
    },
    data=tulli_cv
)
```

```
tulli_mod.initial_values()
```

```
                        componente   value constraint
0                     Check!IDV_V  1.0000          P
1                   repchk!IDV_V -1.0000          P
2                    units!IDV_V  1.0000          P
3  ar1(Column):Row!Residual!SCA_V  1.0000          P
4     ar1(Column):Row!Column!AR_R  0.8315          P
```

```
tulli_mod.summary()['varcomp']
```

```
                                    gamma     sigma  std.error bounds  %ch
Check!IDV_V                      0.480110  0.022290   0.009162   P  4   -2
repchk!IDV_V                    -0.480110 -0.022290   0.009162   C  4    2
units!IDV_V                      2.595457  0.120497   0.009396      P    0
ar1(Column):Row!Residual!SCA_V   1.000000  0.046426   0.010941      P    0
ar1(Column):Row!Column!AR_R      0.870328  0.870328   0.046894      P    0
```

4.6 Estimates from functions of variance components

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

For example, using the data set `pheno_wheat` we fit a Multi-Environment Trial (MET) model and the `vpredict` argument is used to construct expressions related to genetic variance, residual average, standard deviation of the residual average, and heritability, within a Python dictionary, as shown below:

```
met = pd.read_csv("./data/pheno_wheat.csv", sep=",", header=0)
met = define_categorical_variables(met, ["location", "rep", "gen"])

asr_met = asreml(
  response="yield",
    fixed="1 + location",
    random="location:rep + gen",
    residual="dsum(units | location)",
    vpredict={
        "genetic_variance": "V2",
        "residual_average": "(V3 + V4 + V5 + V6 + V7 + V8 + V9 + V10) / 8",
        "sd_residual_average": "SQRT((V3 + V4 + V5 + V6 + V7 + V8 + V9 + V10) /
        8)",
        "heritability": "V2 / (V2 + (V3 + V4 + V5 + V6 + V7 + V8 + V9 + V10)/8)"},
    data=met
)
```

The table with the results is generated by the `.vpredict` method, as follows:

```
asr_met.vpredict()
```

	estimate	std.error
genetic_variance	51.484108	5.075109
residual_average	91.053117	2.975554
sd_residual_average	9.542176	0.155916
heritability	0.361198	0.024098

Chapter 5

Specifying variance structures using genetic and other relationships

5.1 Introduction

In many situations there are relationships between individuals or experimental units, and we expect the variance matrix of the individual effects to be proportional to their relationship matrix. The relationship matrix may arise from spatial, temporal, genetic or other relationships. The previous chapter has discussed models motivated by spatial and temporal relationships. This chapter describes how to specify models with general relationship or inverse relationship matrices, and in addition describes methods to efficiently analyse some common genetic additive relationship matrices that depend on pedigree relationships. ASReml includes a variance model function `vm()` which allows the use of an inverse relationship matrix or one generated from a pedigree (using `ainverse()`).

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

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

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

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

5.2 Pedigree and the numerator relationship matrix

5.2.1 Pedigree objects

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

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

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

```
harvey_ped = pd.read_csv("./data/harvey_ped.csv", skipinitialspace=True)
print(harvey_ped)
```

```
      Calf    Sire  Dam
0   Sire_1       0    0
1   Sire_2       0    0
2   Sire_3       0    0
3   Sire_4       0    0
4   Sire_5       0    0
..     ...     ...  ...
69     161  Sire_9    0
70     162  Sire_9    0
71     163  Sire_9    0
72     164  Sire_9    0
73     165  Sire_9    0

[74 rows x 3 columns]
```

5.2.2 Genetic groups

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

```
harveyg_ped = pd.read_csv("./data/harveyg_ped.csv", skipinitialspace=True)
print(harveyg_ped)
```

```
      Calf    Sire Dam
0       G1       0   0
1       G2       0   0
2       G3       0   0
3   Sire_1      G1  G1
4   Sire_2      G1  G1
..     ...     ...  ..
72     161  Sire_9  G3
73     162  Sire_9  G3
74     163  Sire_9  G3
75     164  Sire_9  G3
76     165  Sire_9  G3

[77 rows x 3 columns]
```

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

5.2.3 Additional qualifiers for numerator relationships

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

- Specifying the level of selfing or inbreeding of an individual.
- Incorporating a column with the gender of an individual. This is relevant to form a relationship matrix for the X chromosome.
- Indicating genetic groups in the pedigree (as shown before).
- Incorporating partial selfing (as a probability).
- Allows for the specification of the inbreeding coefficient for base individuals.
- It can consider the male parent of the female parent (maternal grand-sire) rather than the female parent.

5.3 Specifying relationship and inverse relationship matrices

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

```
harvey_ainv = ainverse(harvey_ped)
print(harvey_ainv['inverse'])
```

```
     rowid  colid      vals
0        1      1  3.666667
1        2      2  3.666667
2        3      3  2.666667
3        4      4  3.666667
4        5      5  3.333333
..     ...    ...       ...
134     72     72  1.333333
135     73      9 -0.666667
136     73     73  1.333333
137     74      9 -0.666667
138     74     74  1.333333

[139 rows x 3 columns]
```

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

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

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

```
harvey = pd.read_csv("./data/harvey.csv", skipinitialspace=True)

harvey = define_categorical_variables(harvey,
                variables=["Calf", "Sire", "Line", "ageOfDam"])
```

```
harvey_ai = ainverse(harvey_ped)
adg0_asr = asreml(
  response="y3",
  fixed="Line",
  random="vm(Calf, harvey_ai)",
  residual="units",
  data={"data": harvey, "harvey_ai": harvey_ai}
)
```

And the variance components are given by:

```
adg0_asr.variance_components()
```

```
                             gamma       sigma   std.error bounds  %ch
vm(Calf,harvey_ai)!GRM_V  1.828628  499.511508  500.533747      P    0
units!Residual!SCA_V      1.000000  273.161938  410.020332      P    0
```

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

```
adg0_asr.coefficients()
```

```
                             estimate  standard_error
Line_1                       0.000000        0.000000
Line_2                      14.071100       13.380034
Line_3                      -5.904215       11.270316
mu                         240.662850        8.560585
vm(Calf,harvey_ai)_Sire_1   11.025278       17.446219
...                               ...             ...
vm(Calf,harvey_ai)_161       9.147409       14.286363
vm(Calf,harvey_ai)_162      21.870626       14.286363
vm(Calf,harvey_ai)_163      10.304065       14.286363
vm(Calf,harvey_ai)_164      13.774034       14.286363
vm(Calf,harvey_ai)_165       7.990753       14.286363

[78 rows x 2 columns]
```

This is an example of an individual animal model (I) estimating additive variance, σ^2_A, and an animal model residual, σ^2_{eI}. In this example we see the estimate of σ^2_A is 499.51 and the estimate of σ^2_{eI} is 273.16. The resulting variance matrix has terms $\sigma^2_A + \sigma^2_{eI}$, the phenotypic variance, in the diagonal elements. The only non-diagonal terms (1/4) in the relationship matrix occur between calves with the same sire (half-sibs) and so their covariance is $(1/4)\sigma^2_A$.

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

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

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

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

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

```
adg1_asr = asreml(
  response="y3",
  fixed="Line",
  random="Sire",
  residual="units",
  data=harvey
)
```

```
adg1_asr.variance_components()
```

```
                         gamma       sigma   std.error bounds  %ch
Sire!IDV_V            0.192774  124.879971  125.128841      P    0
units!Residual!SCA_V  1.000000  647.806494  122.326549      P    0
```

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

```
adg1_asr.coefficients()
```

```
               estimate  standard_error
Line_1         0.000000        0.000000
Line_2        14.071100       13.380058
Line_3        -5.904217       11.270336
mu           240.662852        8.560601
Sire_Sire_1    5.512653        8.723130
Sire_Sire_2   -8.819284        8.723130
Sire_Sire_3    3.306631        9.013478
Sire_Sire_4   -4.009273        9.382863
Sire_Sire_5    4.009273        9.382863
Sire_Sire_6    4.241243        8.565172
Sire_Sire_7    0.222253        8.300560
Sire_Sire_8  -13.482070        8.419124
Sire_Sire_9    9.018574        8.300560
```

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

```
round(adg0_asr.summary(display = False)['loglik'], 3)
```

```
-238.831
```

```
round(adg1_asr.summary(display = False)['loglik'], 3)
```

```
-238.831
```

We see the final `LogLik` is the same in the two models (-238.831) and the E-BLUPs for sires in the individual animal model are twice those in the sire model. The estimate of σ^2_S is 124.88 and the estimate of σ^2_{eS} is 647.81. These convert to give estimates of $\sigma^2_A = 4 \times \sigma^2_S = 4 \times 124.88 = 499.52$ and $\sigma^2_{eI} = \sigma^2_{eS} - 3 \times \sigma^2_S = 647.81 - 3 \times 124.88 = 273.17$ which agree well with the estimates derived from the animal model of $\sigma^2_A = 499.51$ and $\sigma^2_{eI} = 273.16$.

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

```
sire_giv = pd.DataFrame({
    'rowid': range(1, 10),
    'colid': range(1, 10),
    'vals': [0.5] * 9
})

sire_uid = ["Sire_" + str(i) for i in range(1, 10)]

ainv = {"inverse": sire_giv, "ids": sire_uid}

adg2_asr = asreml(
  response="y3",
  fixed="Line",
  random="vm(Sire, ainv)",
  residual="units",
  data={"data": harvey, "ainv": ainv}
)
```

```
adg2_asr.variance_components()
```

```
                          gamma       sigma   std.error bounds  %ch
vm(Sire,ainv)!GRM_V    0.096388   62.442995   62.529141      P    0
units!Residual!SCA_V   1.000000  647.831816  122.333711      P    0
```

In this case $\sigma^2_{S2} = 62.44$, which is in good agreement with the converted value $0.5 \times \sigma^2_S = 124.88/2 = 62.44$ from the previous analysis.

Chapter 6

Prediction from the linear model

6.1 Introduction

Prediction is the process of forming a linear function of the vector of fixed and/or random effects in the linear model to obtain an estimated or predicted value for a quantity of interest. It is primarily used for predicting tables of adjusted means (or predicted margins). If the table is based on a subset of the explanatory variables then the other variables need to be accounted for. It is usual to form a predicted value either at specified values of the remaining variables, or averaging over them in some way.

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

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

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

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

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

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

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

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

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

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

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

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

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

6.2 Estimating means: the `predict` method

The `predict` method is detailed in the help by typing `help(asreml)`. A simple example is the prediction of variety means from fitting model 1b (Section 4.1) to the NIN field trial data. Recall that a randomized block model with random replicate effects is fitted to this data by:

```
rcb_asr = asreml(
  response="yield",
  fixed="Variety",
  random="idv(Rep)",
  residual="idv(units)",
  predict={"classify": "Variety"},
  options={"na_action": {"x": "include","y": "include"}},
  data=nin89)
```

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

```
rcb_pv = rcb_asr.predict()
```

The first few lines of predictions are:

```
print(rcb_pv.predict.head(10))
```

```
      Variety  Pred Values      s.e. Ecode
0    ARAPAHOE      29.4375  3.855683   E
1       BRULE      26.0750  3.855683   E
2    BUCKSKIN      25.5625  3.855683   E
3     CENTURA      21.6500  3.855683   E
4   CENTURK78      30.3000  3.855683   E
5    CHEYENNE      28.0625  3.855683   E
6        CODY      21.2125  3.855683   E
7        COLT      27.0000  3.855683   E
8        GAGE      24.5125  3.855683   E
9   HOMESTEAD      27.6375  3.855683   E
```

6.2.1 The prediction process

Predictions are formed as an extra process in the final iteration of ASReml, and the method `predict` can be used to obtain the predictions. Additional options for `predict` can be set in the predict argument of `asreml()`.

By default, factors are predicted at each level, simple covariates are predicted at their overall mean and covariates used as a basis for splines or orthogonal polynomials are predicted at their design points.

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

1. Including the variables in the *classify* set and specifying any non-default values at which predictions are to be made by using the `levels` key (optional).

```
asreml(
  ...
  predict={"classify": "variable_name", "levels": "variable_levels"},
  ...
)
```

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

```
asreml(
  ...
  predict={"classify": "variable_name", "present": "explanatory_variable"},
  ...
)
```

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

```
asreml(
  ...
  predict={"classify": "variable_name", "weights": [-1,0,1]},
  ...
)
```

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

For example, using the `met` data set, we can fit the model where the predict arguments puts `check:gen` in the classify set and `"{check": 1}` in the levels set. Consequently, the hyper-table is formed from model terms `location` and `at(check, 2):gen` but ignoring all terms in `at(location):ibk`, and then averaging across sites to produce overall test genotype predictions.

```
obj_asr = asreml(
  response="yield",
  fixed="location + at(check, 2):gen",
  random="at(location):ibk + at(check, 1):gen",
  residual="idv(units)",
  predict={"classify": "check:gen", "levels": {"check": 1}},
  data=met
)
```

```
Note     : 2691 records retained of 2700 read
Warning  : 9 observations were dropped due to missing values in response variables
Notice   : specify Sigma parameterization via the options
Note     : forming 635 equations: 282 dense
Note     : initial updates will be shrunk by factor 0.316
Note     : this job uses all of the 12 processor threads
                                          time
       LogLik       Sigma2     DF        (seconds)
  1   -7683.9370   100.4543     2680       0.01 (2 components constrained)
  2   -7632.6144    95.4405     2680       0.02
  3   -7591.9771    89.5553     2680       0.01
  4   -7572.9264    85.5133     2680       0.01
  5   -7568.2097    83.2767     2680       0.01
  6   -7568.0417    82.8481     2680       0.01
  7   -7568.0413    82.8297     2680       0.01
  8   -7568.0413    82.8296     2680       0.02
Notice   : 271 singularities detected in design matrix
```

```
print(obj_asr.predict(display=False).predict.head(n=10))
```

```
          gen  Pred Values      s.e. Ecode
0     Camelot    61.625691  6.774019   E
1     Freeman    61.625691  6.774019   E
2  GOODSTREAK    61.625691  6.774019   E
3     NE16401    56.804340  2.810215   E
4     NE16402    75.944544  2.807791   E
5     NE16403    58.403137  2.807875   E
6     NE16404    57.841078  2.810141   E
7     NE16405    64.612155  2.807899   E
8     NE16406    71.658030  2.808182   E
9     NE16407    50.284931  2.808086   E
```

Note: Controlling display formatting with `pandas.option_context()`

When calling `obj_asr.predict()`, the output is returned as a `pandas` object whose rendering obeys the package's global display settings. The `pandas.option_context` method allows users to set one or more pandas options temporarily within a controlled block.

```
with pd.option_context("display.max_colwidth", None, "display.precision", 3):
    obj_asr.predict()
```

Inside this context, two options are adjusted. Setting `display.max_colwidth` to `None` removes the default truncation of long string cells, ensuring that full text values are printed in their entirety rather than being cut off with an ellipsis. Setting `display.precision` to `3` limits floating-point numbers to three decimal places. Once the `with` block exits, `pandas` automatically restores the previous display settings. For all available options use pandas.describe_option().

6.3 Aliasing

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

1. Linear dependencies among model terms due to over-parameterization of the model.
2. No data present for some factor combinations so that the corresponding effects cannot be estimated.
3. Linear dependencies due to other, usually unexpected, structure in the data.

The first type of aliasing is imposed by the parameterization chosen and can be determined from the model. The second type of aliasing can be detected when setting up the design matrix for parameter estimation (which may require revision of imposed constraints). The third type can then be detected during the absorption of the mixed model equations. Dependencies (aliasing) can be dealt with in several ways and `asreml()` checks that predictions are of estimable functions in the sense defined by Searle (1971) (p. 160) and are invariant to the constraint method used.

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

6.4 Further examples

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

```
nin89_asr = asreml(
  response="yield",
  fixed="Variety",
  random="Rep",
  residual="idv(units)",
  options={"na_action": {"x": "include"}},
  predict={"classify": "Variety"},
  data=nin89
)
```

```
nin89_asr.predict()
```

```
     Variety  Pred Values      s.e. Ecode
0   ARAPAHOE      29.4375  3.855683   E
1      BRULE      26.0750  3.855683   E
2   BUCKSKIN      25.5625  3.855683   E
3    CENTURA      21.6500  3.855683   E
4  CENTURK78      30.3000  3.855683   E
5   CHEYENNE      28.0625  3.855683   E
```

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

```
import numpy as np

nin89['data']['sweight'] = np.random.RandomState(1208).normal(2, 1,
len(nin89['data']))
```

```
nin89_asr = asreml(
  response="yield",
  fixed="Variety + sweight",
  random="Rep",
  residual="idv(units)",
  options={"na_action": {"x": "include"}},
  predict={"classify": "Variety"},
  data=nin89
)
```

```
nin89_asr.predict()
```

```
     Variety  Pred Values      s.e. Ecode
0   ARAPAHOE    29.380175  3.865647   E
1      BRULE    25.995755  3.868730   E
2   BUCKSKIN    25.746415  3.896985   E
3    CENTURA    21.728964  3.868685   E
4  CENTURK78    30.273014  3.863011   E
5   CHEYENNE    27.990496  3.867603   E
```

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

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

```
nin89_asr = asreml(
  response="yield",
  fixed="Variety + sweight + Rep",
  options={"na_action": {"x": "include"}},
  predict={"classify": "Variety:sweight", "levels": {"sweight": 2}},
  data=nin89
)
```

```
nin89_asr.predict()
```

```
     Variety  Pred Values      s.e. Ecode
0   ARAPAHOE    29.401539  3.531986   E
1      BRULE    26.018691  3.534444   E
2   BUCKSKIN    25.750477  3.576295   E
3    CENTURA    21.740554  3.541021   E
4  CENTURK78    30.292202  3.530372   E
5   CHEYENNE    28.012913  3.533513   E
```

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

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

```
checks = met['data'].loc[met['data']['check'] == 1, :]
```

```
obj_asr1 = asreml(
  response="yield",
  fixed="gen",
  random="location:gen",
  residual="dsum(units, location)",
  predict={"classify": "gen"},
  data=checks
)
```

```
obj_asr1.predict()
```

```
  gen  Pred Values      s.e. Ecode
0   4    58.757506  4.003257   E
1   5    76.522028  4.003257   E
2   6    56.775545  4.003257   E
3   7    57.048921  4.003257   E
4   8    65.190784  4.003257   E
5   9    74.050875  4.003257   E
```

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

```
obj_asr2 = asreml(
  response="yield",
  fixed="gen",
  random="location:gen",
  residual="dsum(units, location)",
  predict={"classify": "gen", "weights": {"location": [1/3, 1/3, 1/3, 0, 0, 0, 0,
0]}},
  data=checks
)
```

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

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

```
owt = pd.read_csv("./data/wether.csv")
owt = define_categorical_variables(owt, variables=["Team", "Year"])
```

```
wether_asr = asreml(
  response=["gfw", "fdiam"],
  fixed="trait + trait:Year",
  random="us(trait):Team",
  residual="id(units):us(trait)",
  predict={"classify": "trait:Year"},
  data=owt
)
```

```
wether_asr.predict()
```

```
  Year  trait  Pred Values      s.e. Ecode
0    1    gfw     7.158462  0.109710   E
1    1  fdiam    20.883956  0.214926   E
2    2    gfw     7.034856  0.109177   E
3    2  fdiam    22.231560  0.213572   E
4    3    gfw     8.204947  0.109208   E
5    3  fdiam    23.587171  0.213652   E
```

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

Chapter 7

Examples

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

7.1 Split-plot design

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

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

1. the stratification of the experimental units, that is the *blocks*, *whole-plots*, and *sub-plots*, and
2. the treatment structure that is superimposed on the experimental material.

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

The data can be accessed directly from the file `oats_mod.csv`.

```
oats = pd.read_csv("./data/oats_mod.csv")
print(oats)
```

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

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

```
    Blocks Nitrogen  Subplots      Variety  ...  Column  Row  nrate  yield_mv
0        1  0.6_cwt         1    Marvellous  ...       1    1    NaN     156.0
1        1  0.4_cwt         2    Marvellous  ...       2    1    NaN     118.0
2        1  0.2_cwt         3    Marvellous  ...       1    2    NaN     140.0
3        1    0_cwt         4    Marvellous  ...       2    2    NaN     105.0
4        1    0_cwt         1       Victory  ...       1    3    NaN     111.0
..     ...      ...       ...           ...  ...     ...  ...    ...       ...
67       6  0.2_cwt         4  Golden_rain  ...       4   16    0.2       NaN
68       6    0_cwt         1    Marvellous  ...       3   17    0.0      97.0
69       6  0.2_cwt         2    Marvellous  ...       4   17    0.2      99.0
70       6  0.4_cwt         3    Marvellous  ...       3   18    0.4     119.0
71       6  0.6_cwt         4    Marvellous  ...       4   18    0.6     121.0

[72 rows x 10 columns]
```

```
oats = define_categorical_variables(oats,
    variables=["Blocks", "Nitrogen", "Subplots", "Variety", "Wplots", "Column",
"Row"])
```

The first seven columns are converted to factors as they describe the stratification, or experiment design, and applied treatments. The standard split-plot analysis is achieved by fitting terms `Blocks` and `Blocks:Wplots` as random effects. It is not necessary to specify `Blocks:Wplots:Subplots` as these three factors uniquely define the experimental units. The fixed effects include the main effects

of both `Variety` and `Nitrogen` and their interaction.

```
oats_asr = asreml(
  response="yield",
  fixed="Variety + Nitrogen + Variety:Nitrogen",
  random="Blocks + Blocks:Wplots",
  residual="units",
  options={"wald": {"den_df": "none", "ss_type": "incremental"}},
  data=oats
)
```

The variance components are:

```
oats_asr.variance_components()
```

```
                          gamma       sigma   std.error bounds  %ch
Blocks!IDV_V           1.211164  214.490590  168.660373      P    0
Blocks:Wplots!IDV_V    0.598937  106.068555   67.882005      P    0
units!Residual!SCA_V   1.000000  177.094602   37.336008      P    0
```

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

```
oats_asr.wald()
```

```
                   NumDF       F-inc
mu                     1  245.312491
Variety                2    1.485246
Nitrogen               3   37.683249
Variety:Nitrogen       6    0.302804
```

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

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

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

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

```
oats_wld = asreml(
  response="yield",
  fixed="Variety + Nitrogen + Variety:Nitrogen",
  random="Blocks + Blocks:Wplots",
  residual="units",
  options={"wald": {"den_df": "numeric", "ss_type": "conditional"}},
  predict = [{"classify": "Variety"}, {"classify": "Nitrogen"}, {"classify":
"Variety:Nitrogen"}],
  data=oats
)
```

```
oats_wld.wald()
```

```
                  NumDF  DenDF       F-inc       F-con  M         P-con
mu                    1    5.0  245.141045  245.141045  .  1.931822e-05
Variety               2   10.0    1.485340    1.485340  A  2.723869e-01
Nitrogen              3   45.0   37.685645   37.685645  A  2.457701e-12
Variety:Nitrogen      6   45.0    0.302824    0.302824  B  9.321988e-01
```

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

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

```
oats_pv = oats_wld.predict()
```

```
oats_pv[0].predict
```

```
       Variety  Pred Values      s.e. Ecode
0  Golden_rain   104.500000  7.797538   E
1   Marvellous   109.791667  7.797538   E
2      Victory    97.625000  7.797538   E
```

```
oats_pv[1].predict
```

```
  Nitrogen  Pred Values      s.e. Ecode
0  0.2_cwt    98.888889  7.174709   E
1  0.4_cwt   114.222222  7.174709   E
2  0.6_cwt   123.388889  7.174709   E
3    0_cwt    79.388889  7.174709   E
```

The latter returns an object `oats_pv` . Here, `Pred Values` contains the predicted means for the term defined in the `classify` (`Variety:Nitrogen` here):

```
oats_pv[2].predict
```

```
        Variety Nitrogen  Pred Values      s.e. Ecode
0   Golden_rain  0.2_cwt    98.500000  9.106976   E
1   Golden_rain  0.4_cwt   114.666667  9.106976   E
2   Golden_rain  0.6_cwt   124.833333  9.106976   E
3   Golden_rain    0_cwt    80.000000  9.106976   E
4    Marvellous  0.2_cwt   108.500000  9.106976   E
5    Marvellous  0.4_cwt   117.166667  9.106976   E
6    Marvellous  0.6_cwt   126.833333  9.106976   E
7    Marvellous    0_cwt    86.666667  9.106976   E
8       Victory  0.2_cwt    89.666667  9.106976   E
9       Victory  0.4_cwt   110.833333  9.106976   E
10      Victory  0.6_cwt   118.500000  9.106976   E
11      Victory    0_cwt    71.500000  9.106976   E
```

7.2 Unbalanced nested design

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

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

Table 7.2: Rat data: ANOVA decomposition

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

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

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

```
rats = pd.read_csv("./data/rats.csv", skipinitialspace=True)
print(rats)
```

```
     Dose Sex  littersize  dam  pup  weight
0       C   M          12    1    1    6.60
1       C   M          12    1    2    7.40
2       C   M          12    1    3    7.15
3       C   M          12    1    4    7.24
4       C   M          12    1    5    7.10
..    ...  ..         ...  ...  ...     ...
317  High   M           9   27    5    5.69
318  High   M           9   27    6    6.36
319  High   F           9   27    7    5.93
320  High   F           9   27    8    5.74
321  High   F           9   27    9    5.74

[322 rows x 6 columns]
```

```
rats = define_categorical_variables(rats,variables=["Dose","Sex","dam","pup"])
```

```
rats_asr = asreml(
  response="weight",
  fixed="littersize + Dose + Sex + Dose:Sex",
  random="idv(dam)",
  predict={"classify": "Dose:Sex"},
  residual="units",
  options={"wald": {"den_df": "numeric", "ss_type": "conditional"}},
  data=rats
)
```

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

```
rats_asr.trace(display = False).iloc[[0,5,6,7],[0, 1, 2]]
```

The tables of estimated variance parameters and Wald tests are:

```
rats_asr.variance_components()
```

	gamma	sigma	std.error	bounds	%ch
idv(dam)!IDV_V	0.586674	0.096977	0.033184	P	0
units!Residual!SCA_V	1.000000	0.165300	0.013670	P	0

```
rats_asr.wald()
```

	NumDF	DenDF	F-inc	F-con	M	P-con
mu	1	33.945691	9049.480033	1099.203032	.	0.000000e+00
littersize	1	31.408385	27.991448	46.248596	B	1.202798e-07
Dose	2	23.947055	12.146232	11.506757	A	3.147515e-04
Sex	1	299.769898	57.956627	57.956627	A	3.512746e-13
Dose:Sex	2	302.064977	0.398391	0.398391	B	6.717519e-01

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

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

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

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

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

```
rats_asr2 = asreml(
  response="weight",
  fixed="littersize + Sex + Dose",
  random="idv(dam)",
  residual="units",
  options={"wald": {"den_df": "numeric", "ss_type": "conditional"}},
  data=rats
)
```

```
rats_asr2.variance_components()
```

```
                          gamma     sigma  std.error bounds  %ch
idv(dam)!IDV_V         0.595157  0.097918   0.033414      P    0
units!Residual!SCA_V   1.000000  0.164524   0.013561      P    0
```

```
rats_asr2.wald()
```

```
             NumDF       DenDF        F-inc        F-con  M         P-con
mu               1   33.899689  8981.213224  1093.018043  .  0.000000e+00
littersize       1   31.387677    27.847936    46.428633  A  1.162980e-07
Sex              1  301.694404    59.536523    58.267632  A  3.032019e-13
Dose             2   23.984024    11.417183    11.417183  A  3.283881e-04
```

Note that the variance parameters are re-estimated, though there is little change from the previous analysis.

The impact of (wrongly) dropping `Dam` from this model is shown below:

```
rats_asr3 = asreml(
  response="weight",
  fixed="littersize + Dose + Sex",
  residual="units",
  options={"wald": {"den_df": "numeric", "ss_type": "conditional"}},
  data=rats
)
```

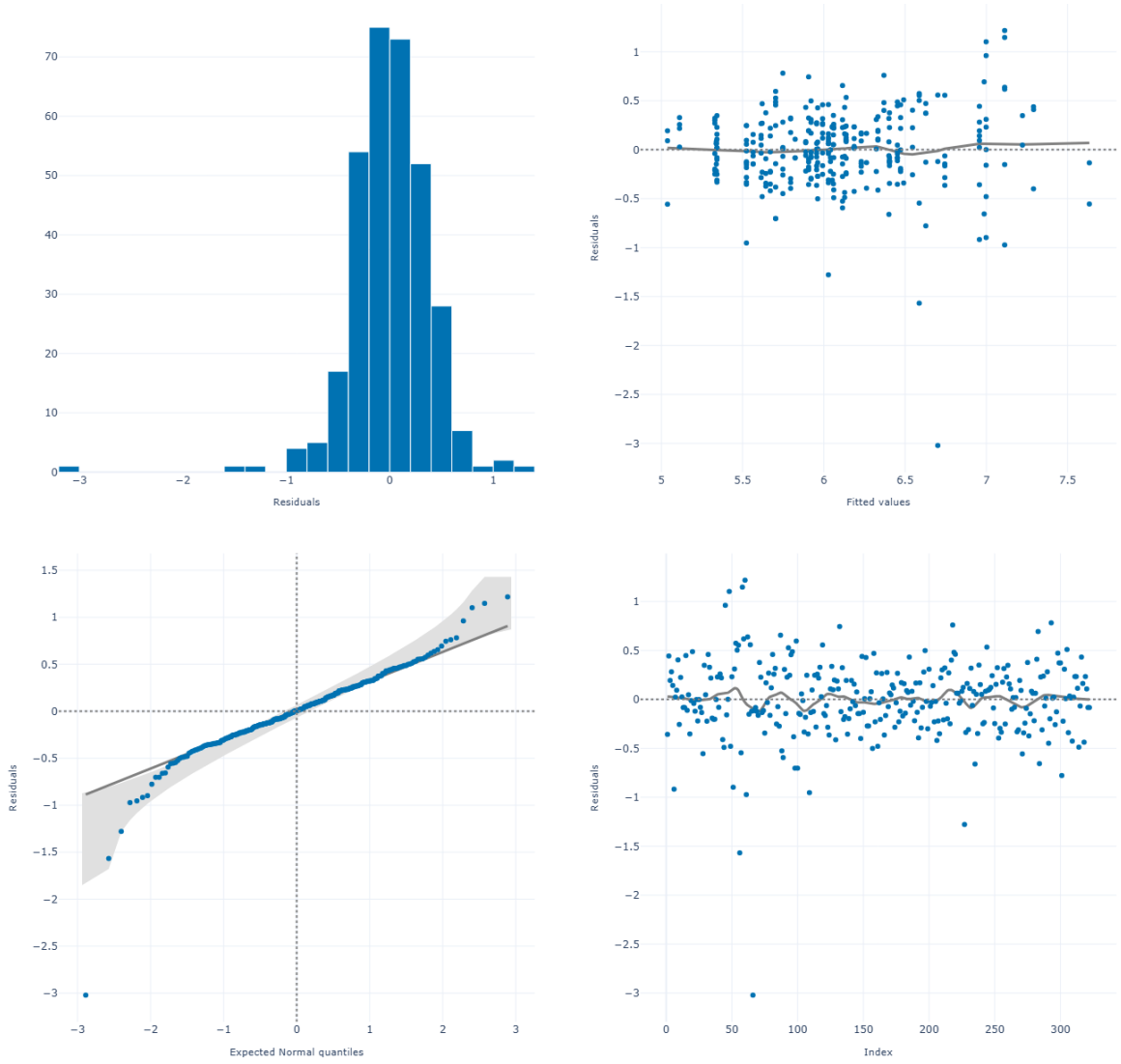

Figure 7.1: Residual plot for the rat data

```
rats_asr3.wald()
```

```
             NumDF  DenDF         F-inc        F-con  M     P-con
mu               1  317.0  47077.305168  3309.417029  .  0.000000
littersize       1  317.0     68.475843   146.500700  A  0.000000
Dose             2  317.0     60.990048    58.431149  A  0.000000
Sex              1  317.0     24.517462    24.517462  A  0.000001
```

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

7.3 Sources of variability in unbalanced data

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

A total of 64 regulators were tested at four *testing stations* (`Teststat`). The voltage for individual regulators was set at a total of 10 *setting stations* (`Setstat`). A variable number of regulators (from 4 to 8) were set at each station. However, each regulator was tested at every testing station. The first few lines of the data set `voltage` are shown below, followed by the `asreml()` function call.

```
voltage = pd.read_csv("./data/voltage.csv", skipinitialspace=True)
print(voltage.head(6))
```

```
   Teststat Setstat  Regulatr  voltage
0         1       A         1     16.5
1         2       A         1     16.5
2         3       A         1     16.6
3         4       A         1     16.6
4         1       A         2     15.8
5         2       A         2     16.7
```

```
voltage = define_categorical_variables(voltage,
              variables=["Setstat","Teststat","Regulatr"])
```

The factor `Regulatr` numbers the regulators within each setting station. Thus the term `Setstat:Regulatr` allows for differential effects of each regulator, while the other terms examine the effects of the setting and testing stations and possible interaction.

```
voltage_asr = asreml(
  response="voltage",
  fixed="mu",
  random="Setstat + Setstat:Regulatr + Teststat + Setstat:Teststat",
  predict={"classify": "Setstat:Regulatr"},
  residual="units",
  data=voltage
)
```

The estimated components of variance are:

```
voltage_asr.variance_components()
```

```
                              gamma         sigma  std.error bounds  %ch
Teststat!IDV_V          6.427519e-02  3.287035e-03   0.003337      P    0
Setstat!IDV_V           2.334162e-01  1.193691e-02   0.008814      P    0
Setstat:Teststat!IDV_V  7.684335e-07  3.929771e-08   0.000000      B    0
Setstat:Regulatr!IDV_V  6.018177e-01  3.077697e-02   0.008453      P    0
units!Residual!SCA_V    1.000000e+00  5.114003e-02   0.005261      P    0
```

The convergence criterion was satisfied, however, the variance component estimate for the `Setstat:Teststat` term has been fixed at the boundary. The default constraint for variance components is to ensure that the REML estimate remains positive. If an update for any variance component results in a negative value then ASReml sets that variance component to a small positive value. If this occurs in subsequent iterations the parameter is fixed at the boundary. The default parameter constraints (Positive for variance components) can be altered (to Unconstrained, for example) by changing the constraint code in the initial value list. This is presented in more detail in Section 3.5.1, though it would not generally be recommended for standard analyses.

Below, we produce a residual plot which indicates two unusual data values (Figure 7.2). These values are successive observations, 210 and 211, respectively, being testing stations 2 and 3 for setting station J, regulator 2. These observations will be retained for consistency with other analyses conducted by Cox and Snell (1981).

Because the estimated variance component for `Setstat:Teststat` is bound at zero, the model omitting this term returns a REML log-likelihood of 203.242 - the same as the REML log-likelihood for the previous model:

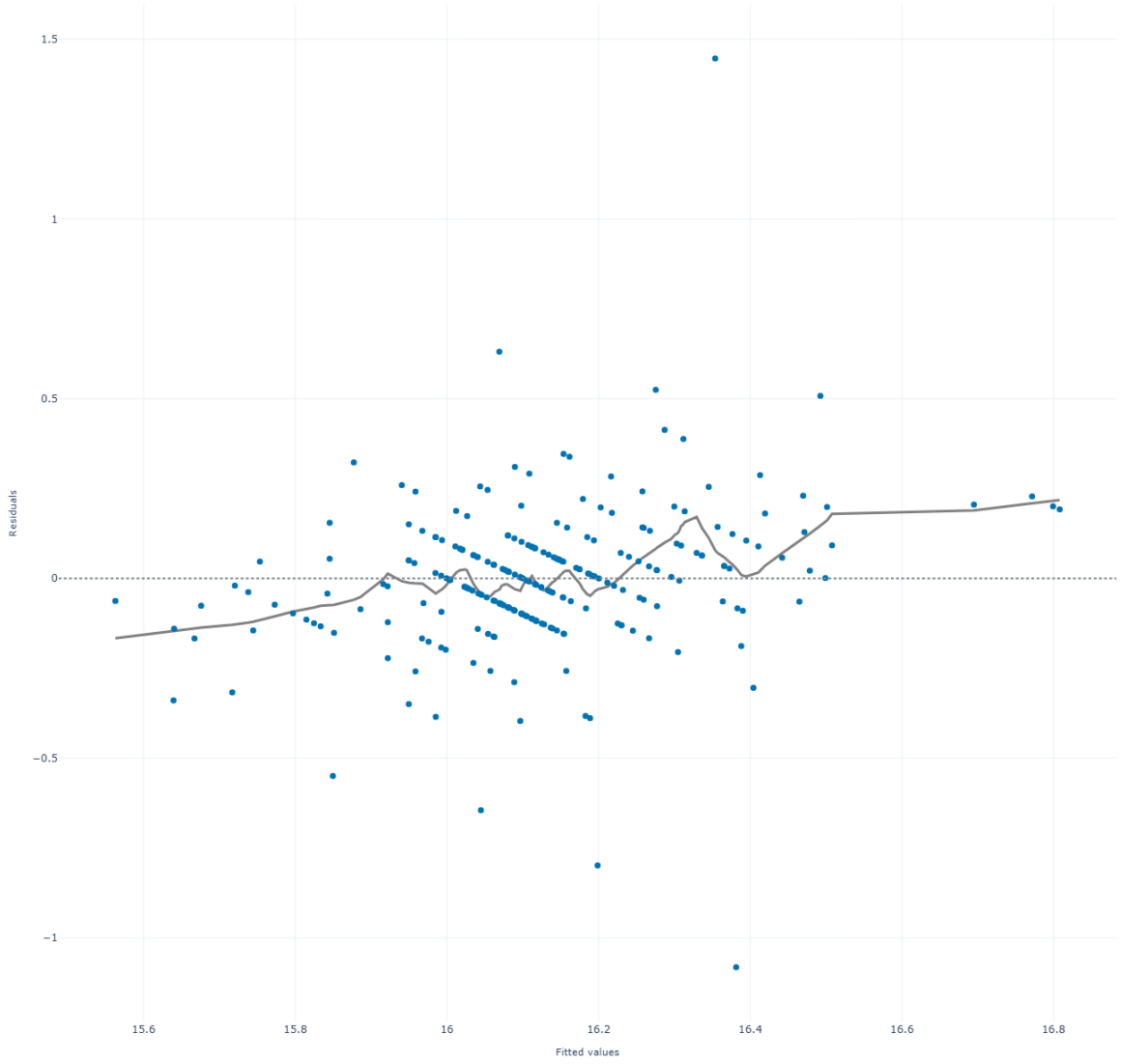

Figure 7.2: Residual vs fitted values for the voltage data

```
voltage_asr2 = asreml(
  response="voltage",
  fixed="mu",
  random="Setstat + Setstat:Regulatr + Teststat",
  predict={"classify": "Setstat:Regulatr"},
  residual="units",
  data=voltage
)
```

A summary of the variance components for the above model is:

```
voltage_asr2.variance_components()
```

```
                          gamma     sigma  std.error bounds  %ch
Teststat!IDV_V          0.064275  0.003287   0.003337      P    0
Setstat!IDV_V           0.233416  0.011937   0.008814      P    0
Setstat:Regulatr!IDV_V  0.601817  0.030777   0.008453      P    0
units!Residual!SCA_V    1.000000  0.051140   0.005261      P    0
```

7.4 Balanced repeated measures

The data for this example comes from an experiment conducted at Rothamsted Experimental Station, UK, by J. Lamptey. It consists of a total of five measurements of height (cm) taken on 14 plants. These plants were either diseased or healthy and were arranged in a glasshouse with a completely randomized design. Plant heights were measured 1, 3, 5, 7 and 10 weeks after the plants were placed in the glasshouse. Therefore, these correspond to repeated measures as each plant was measured five times. There were seven plants in each treatment. The data are illustrated in Figure 7.3.

The following section presents several repeated measures analyses. For some of these it is convenient to arrange the data in a multivariate form, with seven columns containing the plant number, treatment identification and the five height measurements, respectively; while for other analyses, in particular power models, it is necessary to expand the data frame so that the response and a variate for the time of measurement occupy one column each.

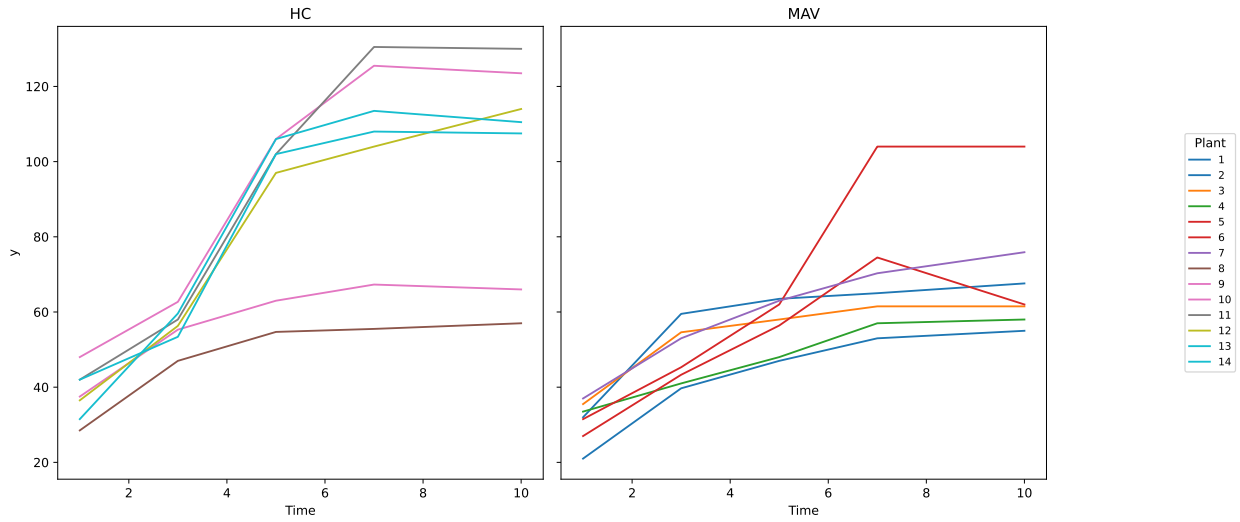

Figure 7.3: Trellis plot of plant height for each of the 14 plants

The data frame `grass` is in *multivariate* form:

```
grass = pd.read_csv("./data/plantm.csv", skipinitialspace=True)
print(grass.head(6))
```

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

```
grass = define_categorical_variables(grass, variables=["tmt"])
```

while `grassUV` is in *univariate* form:

```
grassUV = pd.read_csv("./data/grassUV.csv", skipinitialspace=True)
print(grassUV.head(6))
```

```
   Tmt  Plant  Time HeightID     y
0  MAV      1     1       y1  21.0
1  MAV      1     3       y3  39.7
2  MAV      1     5       y5  47.0
3  MAV      1     7       y7  53.0
4  MAV      1    10      y10  55.0
5  MAV      2     1       y1  32.0
```

```
grassUV = define_categorical_variables(grassUV,
                variables=["Tmt", "HeightID", "Plant", "Time"])
```

The focus is on modelling the error variance for this data. Specifically, we fit the multivariate regression model given by

$$\boldsymbol{Y} = \boldsymbol{DT} + \boldsymbol{E} \tag{7.1}$$

where $\boldsymbol{Y}^{14\times5}$ is the matrix of heights, $\boldsymbol{D}^{14\times2}$ is the trait design matrix, $\boldsymbol{T}^{2\times5}$ is the matrix of fixed effects and $\boldsymbol{E}^{14\times5}$ is the matrix of errors. The heights taken on the same plants will be correlated and so we assume that

$$var(\text{vec}(\boldsymbol{E})) = \boldsymbol{I}_{14} \otimes \boldsymbol{\Sigma} \tag{7.2}$$

where $\boldsymbol{\Sigma}^{5\times5}$ is a symmetric positive definite matrix.

The variance models used in this example for $\boldsymbol{\Sigma}$ are summarized in Table 7.3. They represent some commonly used models for the analysis of repeated measures data (presented by Wolfinger (1996)). Further descriptions of these models can be found in Appendix B. The variance models are fitted by specifying the appropriate special function in the `asreml()` call.

Table 7.3: Summary of variance models fitted to the plant data

model	number of parameters	REML log-likelihood	BIC
uniform	2	-196.88	401.95
power	2	-181.06	370.31
heterogeneous power	6	-170.75	370.16
antedependence (order 1)	9	-160.37	357.51
unstructured	15	-158.04	377.50

The sequence of models given below illustrate some important issues regarding the sort order of the data. In a standard multivariate analysis (data frame `grass`) the response is specified as a matrix and ASReml automatically expands the data frame internally to a univariate form in the order **trait nested within units**. The internal factor `units` is created before this expansion. The data frame `grassUV` has been expanded outside ASReml in the same order, that is **trait nested within experimental units**. In this case ASReml cannot sensibly create a correct `units` factor so a factor defining the experimental units must already exist - in this case the factor `Plant` can be used.

Note that the sort order of the data must correspond to the order of appearance of the factors in the `residual` formula that defines the experimental units. In the case of the one-dimensional power model, the data must be sorted in the order returned by `unique(x)` where `x` is the column in the

data frame containing the distances. In this case ASReml checks the sort order and reports an error if incorrect.

Split-plot in time

The split-plot in time model fitting approach can be implemented four ways:

1. By fitting a random `units` term plus an independent residual using the *multivariate* data frame. This model is also known as Uniform.

```
grass_asr = asreml(
    response=["y1", "y3", "y5", "y7", "y10"],
    fixed="trait + tmt + trait:tmt",
    random="idv(units)",
    residual="id(units):idv(trait)",
    options={"asuv": True},
    data=grass
)
```

2. By specifying a `corv()` variance model for the R structure, again using the *multivariate* data frame.

```
grass_asr1 = asreml(
    response=["y1", "y3", "y5", "y7", "y10"],
    fixed="trait + tmt + trait:tmt",
    residual="id(units):corv(trait)",
    predict={"classify": "trait:tmt"},
    options={"asuv": True},
    data=grass
)
```

As this residual is specified as a correlation matrix the model is fitted on the gamma parameterization. The residual variance is denoted as `S2` in the convergence trace.

3. By fitting `Plant` as a random term plus an independent residual (`Time:Plant`) using the *univariate* data frame.

```
grass_asr1a = asreml(
  response="y",
  fixed="Tmt + Time + Tmt:Time",
  random="idv(Plant)",
  residual="id(Plant):idv(Time)",
  data=grassUV
)
```

4. By specifying a `corv()` variance model for the `Time:Plant` residual term using the *univariate* data.

```
grass_asr1b = asreml(
  response="y",
  fixed="Tmt + Time + Tmt:Time",
  residual="id(Plant):corv(Time)",
  data=grassUV,
  fit_options={"process_stage": "preprocess"}
)
```

The options **1** and **3** are equivalent as are **2** and **4**. The two forms for Σ are given by

$$\begin{aligned} \boldsymbol{\Sigma} &= \sigma_1^2 \boldsymbol{J} + \sigma_2^2 \boldsymbol{I} \qquad \texttt{idv(units)} \qquad (7.3) \\ \boldsymbol{\Sigma} &= \sigma_e^2 \boldsymbol{I} + \sigma_e^2 \rho (\boldsymbol{J} - \boldsymbol{I}) \qquad \texttt{corv()} \qquad (7.4) \end{aligned}$$

It follows that

$$\begin{aligned} \sigma_e^2 &= \sigma_1^2 + \sigma_2^2 \\ \rho &= \frac{\sigma_1^2}{\sigma_1^2 + \sigma_2^2} \end{aligned} \qquad (7.5)$$

Summaries of the outputs from options **1** and **2** (the `asreml()` calls labelled **Uniform** and **Correlation**, respectively) follow. The REML log-likelihood is the same for both models and it is easy to verify that the REML estimates of the variance parameters satisfy (7.5).

```
round(grass_asr.summary(display = False)['loglik'], 4)
```

```
-196.8768
```

```
grass_asr.variance_components()
```

```
                                    gamma       sigma  ...  bounds %ch
idv(units)!IDV_V                 1.263422  159.815655  ...       P   0
id(units):idv(trait)!Residual!SCA_V  1.000000  126.494250  ...       P   0

[2 rows x 5 columns]
```

```
round(grass_asr1.summary(display = False)['loglik'], 4)
```

```
-196.8768
```

```
grass_asr1.variance_components()
```

```
                                        gamma       sigma  ...  bounds %ch
id(units):corv(trait)!Residual!SCA_V  1.000000  286.309905  ...       P   0
id(units):corv(trait)!trait!COR_R     0.558191    0.558191  ...       P   0

[2 rows x 5 columns]
```

Decaying correlation structure

A more plausible model for repeated measures data would allow the correlations to decrease as the lag increases. The simplest model that accommodates this is the first order autoregressive model. However, since the heights are not measured at equally-spaced time points we use the exponential or power model `exp()`. The correlation function is given by:

$$\rho(u) = \phi^{|u|}$$

where u is the time lag (in this case weeks). The code for this model (also known as Power) is presented below.

```
grass_asr2 = asreml(
  response="y",
  fixed="Tmt + Time + Tmt:Time",
  residual="id(Plant):expv(Time)",
  data=grassUV
)
```

The variance parameters from this model are:

```
grass_asr2.variance_components()
```

```
                                       gamma       sigma  ...  bounds %ch
id(Plant):expv(Time)!Residual!SCA_V  1.000000  287.667022  ...       P   0
id(Plant):expv(Time)!Time!EXP_P      0.834076    0.834076  ...       P   0

[2 rows x 5 columns]
```

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

The code below creates a trend plot (Figure 7.4) of residuals against the factors that index the experimental units.

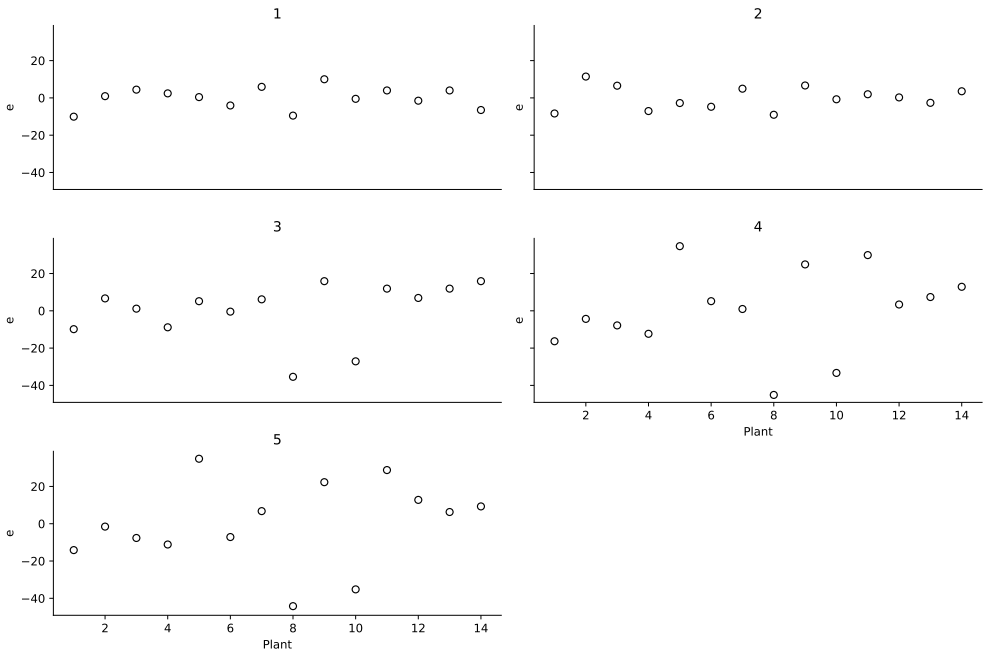

Figure 7.4: Residuals by Plant | Time for the exp() variance model of the plant data

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

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

where $\boldsymbol{D}$ is a diagonal matrix of variances and $\boldsymbol{C}$ is a correlation matrix with elements given by $c_{ij} = \phi^{|t_i - t_j|}$.

The code and parameter estimates for this **Heterogeneous power** model are:

```
grass_asr3 = asreml(
  response="y",
  fixed="Tmt + Time + Tmt:Time",
  residual="id(Plant):exph(Time)",
  data=grassUV,
  fit_options={"process_stage": "preprocess"}
)
```

```
# Set initial values to 60,0.90,60,73,308,434,380
grass_ini = grass_asr3.initial_values(display=True)
```

```
grass_ini["id(Plant):exph(Time)!Residual!SCA_V"]["values"] = 60
grass_ini["id(Plant):exph(Time)!Time_1!EXP_R"]["values"] = 0.90
grass_ini["id(Plant):exph(Time)!Time_1!EXP_V"]["values"] = 60
grass_ini["id(Plant):exph(Time)!Time_2!EXP_V"]["values"] = 73
grass_ini["id(Plant):exph(Time)!Time_3!EXP_V"]["values"] = 308
grass_ini["id(Plant):exph(Time)!Time_4!EXP_V"]["values"] = 434
grass_ini["id(Plant):exph(Time)!Time_5!EXP_V"]["values"] = 380
grass_asr3.fit(initial_values=grass_ini)
```

```
grass_asr3.variance_components()
```

```
                                      gamma        sigma  ...  bounds  %ch
id(Plant):exph(Time)!Residual!SCA_V  1.000000  2525.043247  ...       S    0
id(Plant):exph(Time)!Time_1!EXP_P    0.467608     0.467608  ...       P  143
id(Plant):exph(Time)!Time_1!EXP_V    0.009487    23.954664  ...       ?  -82
id(Plant):exph(Time)!Time_2!EXP_V    0.009487    23.954664  ...       ?  -76
id(Plant):exph(Time)!Time_3!EXP_V    0.009487    23.954664  ...       ? -162
id(Plant):exph(Time)!Time_4!EXP_V    0.009487    23.954664  ...       ? -196
id(Plant):exph(Time)!Time_5!EXP_V    0.081745   206.410203  ...       P  -48

[7 rows x 5 columns]
```

Note that ASReml fixes the scale parameter to 1.0 to ensure that the elements of $\boldsymbol{D}$ are identifiable.

Unstructured correlation structure

The final two models considered are the antedependence model of order 1 and the unstructured model. They consider by default as starting values the gammas of 0.15 for variances and 0.10 for covariances and scale these by 1/2 of the simple variance of the response. This is adequate in many cases (including this example) but we would generally recommend using the REML estimate of $\boldsymbol{\Sigma}$ from a previous model. For example, suitable starting values could be generated from the heterogeneous power model (`grass_asr3`)

The antedependence form models $\boldsymbol{\Sigma}$ by the inverse Cholesky decomposition

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

where $\boldsymbol{D}$ is a diagonal matrix, and $\boldsymbol{U}$ is a unit upper triangular matrix. For an antedependence model of order q, then $l_{ij} = 0$ for $j > i + q - 1$. The antedependence model of order 1 has nine parameters for these data, five in $\boldsymbol{D}$ and four in $\boldsymbol{U}$. The call using the default starting values was shown before.

The code and antedependence parameter estimates follow and appear successively for each time, that is, the element of $\boldsymbol{D}$ and then the row of $\boldsymbol{U}$:

```
grass_asr4 = asreml(
    response=["y1", "y3", "y5", "y7", "y10"],
    fixed="trait + tmt + trait:tmt",
    residual="id(units):ante(trait, 1)",
    options={"asuv": True},
    data=grass
)
```

```
grass_asr4.variance_components()
```

```
                                              sigma  std.error bounds  %ch
id(units):ante(trait,1)!trait,1_1_1!ANTE_U  0.026868   0.011024      P    0
id(units):ante(trait,1)!trait,1_2_1!ANTE_U -0.628348   0.245831      P    0
id(units):ante(trait,1)!trait,1_2_2!ANTE_U  0.037282   0.015475      P    0
id(units):ante(trait,1)!trait,1_3_2!ANTE_U -1.491223   0.586926      P    0
id(units):ante(trait,1)!trait,1_3_3!ANTE_U  0.005996   0.002467      P    0
id(units):ante(trait,1)!trait,1_4_3!ANTE_U -1.280717   0.206890      P    0
id(units):ante(trait,1)!trait,1_4_4!ANTE_U  0.007897   0.003232      P    0
id(units):ante(trait,1)!trait,1_5_4!ANTE_U -0.967777   0.062827      P    0
id(units):ante(trait,1)!trait,1_5_5!ANTE_U  0.039063   0.015948      P    0
```

In this particular case the residual specifies a variance matrix so the sigma parameterization is used for estimation (see Section 2.1.1). The default for multivariate analyses is to use the sigma parameterization.

Finally, the code and estimated components for the unstructured model using default starting values is

```
grass_asr5 = asreml(
    response=["y1", "y3", "y5", "y7", "y10"],
    fixed="trait + tmt + trait:tmt",
    residual="id(units):us(trait)",
    options={"asuv": True},
    data=grass
)
```

```
grass_asr5.variance_components()
```

```
                                          sigma   std.error bounds  %ch
id(units):us(trait)!trait_1_1!US_V   37.226024   15.174440      P    0
id(units):us(trait)!trait_2_1!US_C   23.393310   13.186829      P    0
id(units):us(trait)!trait_2_2!US_V   41.519402   16.933313      P    0
id(units):us(trait)!trait_3_1!US_C   51.651555   31.909574      P    0
id(units):us(trait)!trait_3_2!US_C   61.916197   34.753823      P    0
id(units):us(trait)!trait_3_3!US_V  259.117325  105.216605      P    0
id(units):us(trait)!trait_4_1!US_C   70.809983   45.931457      P    0
id(units):us(trait)!trait_4_2!US_C   57.613387   46.548748      P    0
id(units):us(trait)!trait_4_3!US_C  331.800199  144.287913      P    1
id(units):us(trait)!trait_4_4!US_V  551.496333  223.685272      P    1
id(units):us(trait)!trait_5_1!US_C   73.784402   46.007572      P    0
id(units):us(trait)!trait_5_2!US_C   62.567923   46.734668      P    0
id(units):us(trait)!trait_5_3!US_C  330.844077  143.418905      P    1
id(units):us(trait)!trait_5_4!US_C  533.745371  219.136721      P    1
id(units):us(trait)!trait_5_5!US_V  542.165121  219.905746      P    1
```

The antedependence model of order 1 is clearly the more parsimonious model (see Table 7.3). There is a surprising level of discrepancy between models for the Wald tests (see Table 7.4 below). The main effect of treatment is significant for the uniform and power models. In this case, we have used `{"wald": {"den_df": "numeric", "ss_type" :"conditional"}}`.

Table 7.4: Summary of Wald statistics for fixed effects for the models fitted to the plant data

model	`Tmt` (df = 1)	p-value	`trait:Tmt` (df = 4)	p-value
uniform	9.41	0.0098	5.10	0.0017
power	6.85	0.0236	6.01	0.0005
heterogeneous power	0.00	0.9836	4.32	0.0105
antedependence (order 1)	4.16	0.0634	3.34	0.0384
unstructured	1.72	0.2141	3.34	0.0610

7.5 Spatial analysis of a field experiment

This section illustrates spatial and incomplete block analyses of a field experiment using ASReml. There has been a large amount of interest in developing techniques for the analysis of spatial data both in the context of field experiments and geostatistical data (Cressie 1991; Cullis and Gleeson 1991; Gilmour et al. 1997, to name a few). This example illustrates the analysis of *so-called* regular spatial data, in which the data are observed on a lattice or regular grid. This is typical

of most small-plot designed field experiments. Spatial data are often irregularly spaced, either by design or because of the observational nature of the study. The techniques we present here can be extended for the analysis of irregularly spaced spatial data, though, larger spatial data sets may be computationally challenging, depending on the degree of irregularity or models fitted.

The data used here appears in Gilmour et al. (1995) and is from a field experiment designed to compare the performance of 25 varieties of barley. The experiment was conducted at Slate Hall Farm, UK, in 1976 and was designed as a balanced lattice square with six replicates laid out in a 10×15 rectangular grid. Table 7.5 shows the layout of the experiment and the coding of the replicates and lattice blocks. The first few rows in the data are:

```
shf = pd.read_csv("./data/shf.csv", skipinitialspace=True)
print(shf.head(6))
```

```
   Replicate  colblk  rowblk  variety  yield  row  column
0          1       1       1        1   1003    1       1
1          1       1       2        2   1356    2       1
2          1       1       3        4   1412    3       1
3          1       1       4        3   1239    4       1
4          1       1       5        5   1508    5       1
5          2      11       6       19   1967    6       1
```

```
shf = define_categorical_variables(shf,
             variables=["Replicate","colblk","rowblk","variety","row","column"])
```

Lattice block numbering is typically coded *within* replicates, however, in this example the lattice row and column blocks were both numbered from 1 to 30. The terms in the linear model are therefore simply `rowblk` and `colblk`. The factors `row` and `column` help to identify the spatial layout of the plots.

Three models are considered below: two spatial and the traditional incomplete block design (for comparative purposes). In the first model, we fit a separable first order autoregressive process to the variance structure of the plot errors. Gilmour et al. (1997) suggest this is often a useful model to commence the spatial modelling process. The form of the variance matrix for the plot errors ($\boldsymbol{R}$-structure) is given by

$$\boldsymbol{\Sigma} = \sigma^2(\boldsymbol{\Sigma}_c \otimes \boldsymbol{\Sigma}_r) \tag{7.6}$$

where the $\boldsymbol{\Sigma}_c$ and $\boldsymbol{\Sigma}_r$ are 15×15 and 10×10 correlation matrix functions of the column (ϕ_c) and row (ϕ_r) autoregressive parameters respectively.

The separable autoregressive error model is fitted by:

Table 7.5: Field layout of Slate Hall Farm experiment

Column - Replicate levels															
Row	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	1	1	1	1	1	2	2	2	2	2	3	3	3	3	3
2	1	1	1	1	1	2	2	2	2	2	3	3	3	3	3
3	1	1	1	1	1	2	2	2	2	2	3	3	3	3	3
4	1	1	1	1	1	2	2	2	2	2	3	3	3	3	3
5	1	1	1	1	1	2	2	2	2	2	3	3	3	3	3
6	4	4	4	4	4	5	5	5	5	5	6	6	6	6	6
7	4	4	4	4	4	5	5	5	5	5	6	6	6	6	6
8	4	4	4	4	4	5	5	5	5	5	6	6	6	6	6
9	4	4	4	4	4	5	5	5	5	5	6	6	6	6	6
10	4	4	4	4	4	5	5	5	5	5	6	6	6	6	6
Column - Rowblk levels															
Row	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	1	1	1	1	1	11	11	11	11	11	21	21	21	21	21
2	2	2	2	2	2	12	12	12	12	12	22	22	22	22	22
3	3	3	3	3	3	13	13	13	13	13	23	23	23	23	23
4	4	4	4	4	4	14	14	14	14	14	24	24	24	24	24
5	5	5	5	5	5	15	15	15	15	15	25	25	25	25	25
6	6	6	6	6	6	16	16	16	16	16	26	26	26	26	26
7	7	7	7	7	7	17	17	17	17	17	27	27	27	27	27
8	8	8	8	8	8	18	18	18	18	18	28	28	28	28	28
9	9	9	9	9	9	19	19	19	19	19	29	29	29	29	29
10	10	10	10	10	10	20	20	20	20	20	30	30	30	30	30
Column - Colblk levels															
Row	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
2	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
3	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
4	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
5	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
6	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
7	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
8	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
9	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
10	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30

```
barley1_asr = asreml(
  response="yield",
  fixed="variety",
  residual="ar1v(row):ar1(column)",
  predict={"classify": "variety"},
  options={"wald": {"den_df": "numeric", "ss_type": "conditional"}},
  data=shf
)
```

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

```
round(barley1_asr.summary(display = False)['loglik'], 4)
```

```
-700.3225
```

```
barley1_asr.variance_components()
```

```
                                          gamma         sigma  ...  bounds %ch
ar1v(row):ar1(column)!Residual!SCA_V  1.000000  38756.330404  ...       P   0
ar1v(row):ar1(column)!row!AR_R        0.683767      0.683767  ...       P   0
ar1v(row):ar1(column)!column!AR_R     0.458606      0.458606  ...       P   0

[3 rows x 5 columns]
```

```
barley1_asr.wald()
```

```
          NumDF      DenDF       F-inc       F-con  M         P-con
mu            1  12.786034  851.027562  851.027562  .  4.423129e-13
variety      24  79.995646   13.035909   13.035909  A  0.000000e+00
```

```
variogram(barley1_asr, file_name="./plot/3b-1.pdf")
```

Gilmour et al. (1997) recommend revision of the current spatial model based on the use of diagnostics such as the sample variogram of the residuals. This diagnostic plot is produced by the `variogram()` method, and presented in Figure 7.5. Additional technical details are presented in Section 2.5.3.

The iterative sequence has converged to *row* and *column* correlation parameters of 0.6837 and 0.4586, respectively. The plot size and orientation is not known and so it is not possible to ascertain whether these values are spatially sensible. It is generally found that the closer the plot centroids, the higher the spatial correlation. This is not always the case, and if the highest between-plot correlation relates to the larger spatial distance then this may suggest the presence of extraneous variation (see Gilmour et al. (1997), for example). An extension to this model includes a measurement error or *nugget effect* term, which follows:

```
barley2_asr = asreml(
  response="yield",
  fixed="variety",
  random = "units",
  residual="ar1v(row):ar1(column)",
  predict={"classify": "variety"},
  options={"wald": {"den_df": "numeric", "ss_type": "conditional"}},
  data=shf
)
```

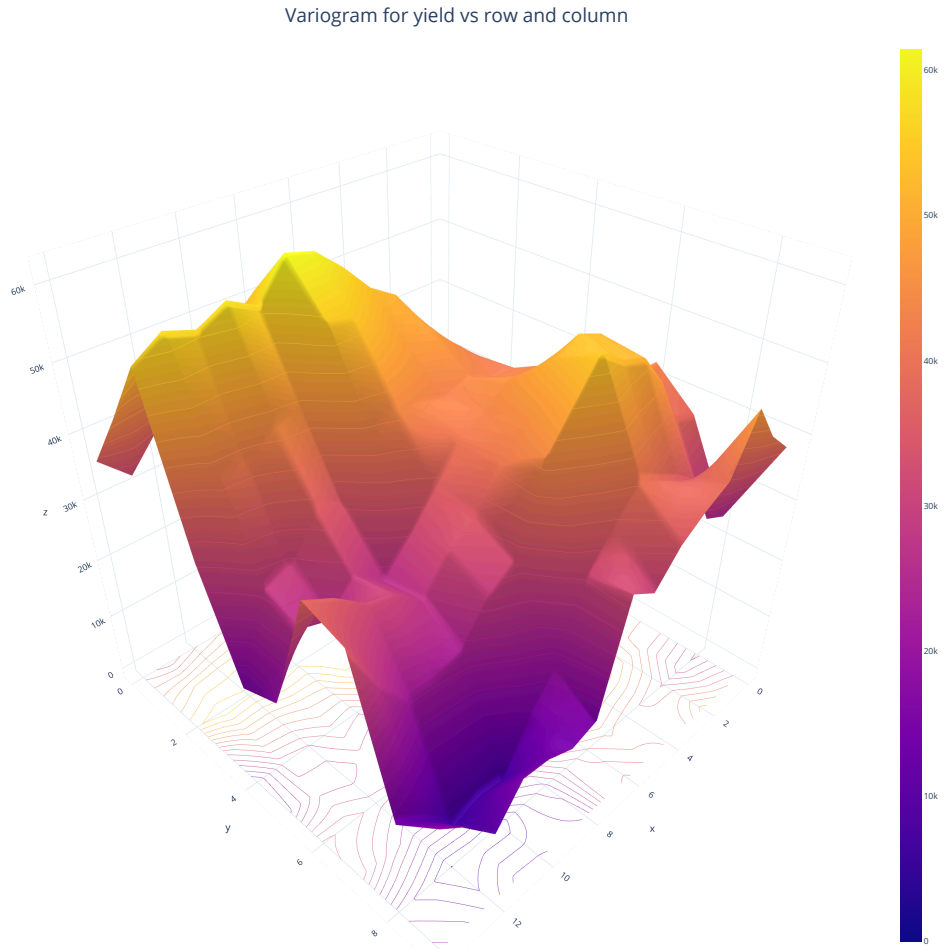

Figure 7.5: Sample variogram of the $AR1$ x $AR1$ model for the Slate Hall data

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

$$\boldsymbol{\Sigma} = \sigma^2(\boldsymbol{\Sigma}_c \otimes \boldsymbol{\Sigma}_r) + \psi \boldsymbol{I}_{150} \tag{7.7}$$

where ψ is the ratio of nugget variance to error variance (σ^2). The results show a significant improvement in the REML log-likelihood with the inclusion of the nugget effect (see Table 7.6).

```
round(barley2_asr.summary(display = False)['loglik'], 4)
```

```
-696.8227
```

```
barley2_asr.variance_components()
```

```
                                          gamma         sigma  ...  bounds %ch
units!IDV_V                            0.106153   4862.047425  ...       P   0
ar1v(row):ar1(column)!Residual!SCA_V   1.000000  45802.433158  ...       P   0
ar1v(row):ar1(column)!row!AR_R         0.843798      0.843798  ...       P   0
ar1v(row):ar1(column)!column!AR_R      0.682695      0.682695  ...       P   0

[4 rows x 5 columns]
```

```
barley2_asr.wald()
```

```
          NumDF      DenDF       F-inc       F-con  M         P-con
mu            1   3.507079  259.846341  259.846341  .  2.050307e-04
variety      24  75.662403   10.211608   10.211608  A  2.997602e-15
```

Finally, the incomplete block analysis (with recovery of inter-block information) is:

```
barley3_asr = asreml(
  response="yield",
  fixed="variety",
  random = "rowblk + colblk",
  residual="units",
  predict={"classify": "variety"},
  options={"wald": {"den_df": "numeric", "ss_type": "conditional"}},
  data=shf
)
```

```
round(barley3_asr.summary(display = False)['loglik'], 4)
```

```
-708.1182
```

```
barley3_asr.variance_components()
```

```
                          gamma         sigma    std.error bounds  %ch
rowblk!IDV_V           1.974324  15882.635385  5124.209948      P    0
colblk!IDV_V           2.085503  16777.024313  5363.649857      P    0
units!Residual!SCA_V   1.000000   8044.594649  1335.972420      P    0
```

```
barley3_asr.wald()
```

```
          NumDF      DenDF        F-inc        F-con  M         P-con
mu            1  47.263274  1892.865156  1892.865156  .  0.000000e+00
variety      24  78.986753     8.846665     8.846665  A  5.950795e-14
```

In general, this variance model is not competitive with the preceding spatial models and this can be noted from the log-likelihood values. The models can be formally compared using the AIC or BIC values, for example. Note that to compare fits using these criteria, the models need to have the same fixed effects (which in this case is only the overall mean), and the smaller the value the better the model.

The Wald statistics for the spatial models (`F-con`) are greater than that for the incomplete block analysis (Table 7.6). We note that the Wald statistic for the spatial model including the nugget effect is smaller than that for the AR1×AR1 model.

Table 7.6: Summary of models fitted to the Slate Hall data

model	REML log-likelihood	AIC	BIC	parameters	F-con	SED
AR1×AR1	-700.32	1406.65	1415.13	3	13.04	59.05
AR1×AR1 + units	-696.82	1401.65	1412.96	4	10.21	60.51
incomplete block	-708.12	1422.23	1430.72	3	08.85	62.04

Finally, the `asreml()` calls included an argument to predict `Variety` means. The first six means are reproduced here for each of the models. The overall SED is the square root of the average variance of difference between the variety means. Note that all variety comparisons have the same SED for the balanced lattice square analysis.

```
barley1_pv = barley1_asr.predict(display=False)
print(barley1_pv.predict.head(n=6))
```

```
  variety  Pred Values       s.e. Ecode
0       1  1257.974668  64.612635   E
1       2  1501.450899  64.976206   E
2       3  1404.987799  64.623898   E
3       4  1412.567109  64.900603   E
4       5  1514.474783  65.586878   E
5       6  1553.437351  64.144181   E

SED      : overall standard error of difference 59.05
```

```
barley2_pv = barley2_asr.predict(display=False)
print(barley2_pv.predict.head(6))
```

```
  variety  Pred Values       s.e. Ecode
0       1  1245.587527  97.855658   E
1       2  1516.232445  97.843924   E
2       3  1403.988119  98.236388   E
3       4  1404.922968  97.984100   E
4       5  1471.625905  98.357311   E
5       6  1521.947596  97.967624   E

SED      : overall standard error of difference 60.51
```

```
barley3_pv = barley3_asr.predict(display=False)
print(barley3_pv.predict.head(6))
```

```
  variety  Pred Values       s.e. Ecode
0       1  1284.373487  54.681556   E
1       2  1549.437700  54.681556   E
2       3  1420.574395  54.681556   E
3       4  1452.114210  54.681556   E
4       5  1534.197636  54.681556   E
5       6  1527.896110  54.681556   E

SED      : overall standard error of difference 62.04
```

7.6 Unreplicated early generation variety trial

This example is a further illustration of the approach to the analysis of field trials presented in the previous section. The data are from an unreplicated field experiment conducted at Tullibigeal in south-western New South Wales, Australia. The trial was an S1 (early stage) wheat variety evaluation trial and consisted of 525 test lines which were randomly assigned to plots in a 67 row $\times$ 10 column array. There was a check variety/line every six plots within each column. That is, the check variety was sown on rows 1, 7, 13, ..., 67 of each column. This variety was numbered 526. A further six replicated commercially available varieties (numbered 527 to 532) were also randomly assigned to plots between three to five plots each. The aim of these trials is to identify and retain the top, say 20%, lines for further testing.

Cullis et al. (1989) considered the analysis of early generation variety trials and presented a one-dimensional spatial analysis which was an extension of the approach developed by Gleeson and Cullis (1987). The test line effects are assumed random, while the check variety effects are considered fixed. This may not be sensible or justifiable for most trials and can lead to inconsistent comparisons between check varieties and test lines. Given the large amount of replication afforded to check varieties there will be very little shrinkage irrespective of the realised heritability.

In the following we assume that the variety effect (including both check, replicated and unreplicated lines) is random. In addition to a one-dimensional analysis, we consider three further spatial models for comparison. The first few rows of the data set `wheat` are presented below:

```
wheat = pd.read_csv("./data/wheat_earlygen.csv", skipinitialspace=True)
print(wheat)
```

```
      yield  weed  Column  Row  Variety
0    2652.0     0       1    1      526
1    2691.0     0       2    1      526
2    2770.0     0       3    1      526
3    2896.0     0       4    1      526
4    2473.0     1       5    1      526
..      ...   ...     ...  ...      ...
665  2042.0     2       6   67      526
666  2296.0     2       7   67      526
667  1795.0     3       8   67      526
668  1727.0     5       9   67      526
669  1728.0     2      10   67      526

[670 rows x 5 columns]
```

```
wheat.sort_values(['Row','Column'])
```

```
wheat = define_categorical_variables(wheat,
               variables=["Row","Column","Variety"])
```

where `Variety`, `Row`, and `Column` are factors, `yield` is the response variate and `weed` is a covariate. Note that the data frame is sorted as *Column* nested within *Row*.

We begin with a one-dimensional spatial model, which assumes the variance model for the plot effects within columns is described by a first order autoregressive process.

```
wheat_asr1 = asreml(
  response="yield",
  fixed="weed",
  random="idv(Variety)",
  residual="ar1v(Row):id(Column)",
  options={"na_action": {"x": "include","y": "include"}},
  data=wheat)
```

The REML log-likelihood and variance components from the fit are:

```
round(wheat_asr1.summary(display = False)['loglik'], 4)
```

```
-4239.88
```

```
wheat_asr1.variance_components()
```

```
                                          gamma          sigma  ...  bounds %ch
idv(Variety)!IDV_V                     0.959421   82788.464275  ...       P   0
ar1v(Row):id(Column)!Residual!SCA_V    1.000000   86289.992261  ...       P   0
ar1v(Row):id(Column)!Row!AR_R          0.672284       0.672284  ...       P   0

[3 rows x 5 columns]
```

The REML estimate of the autoregressive parameter indicates substantial within-column spatial correlation.

A two-dimensional spatial model is fitted with:

```
wheat_asr2 = asreml(
  response="yield",
  fixed="weed",
  random="idv(Variety)",
  residual="ar1v(Row):ar1(Column)",
  options={"na_action": {"x": "include","y": "include"}},
  data=wheat)
```

```
round(wheat_asr2.summary(display=False)['loglik'], 4)
```

```
-4233.6478
```

```
wheat_asr2.variance_components()
```

```
                                          gamma         sigma  ...  bounds %ch
idv(Variety)!IDV_V                     1.060320  88106.415471  ...       P   0
ar1v(Row):ar1(Column)!Residual!SCA_V   1.000000  83094.178360  ...       P   0
ar1v(Row):ar1(Column)!Row!AR_R         0.685321      0.685321  ...       P   1
ar1v(Row):ar1(Column)!Column!AR_R      0.285859      0.285859  ...       P   0

[4 rows x 5 columns]
```

The change in REML log-likelihood is significant ($\chi^2_1 = 12.46, P < 0.001$) with the inclusion of the autoregressive parameter for `Column`. The sample variogram of the residuals for the $AR1 \times AR1$ model (see Figure 7.6) indicates a linear drift from column 1 to column 10. We will include a linear regression coefficient `lin(Column)` in the model to account for this. But this will not make the REML log-likelihood values comparable as we have different fixed effects.

```
variogram(wheat_asr2, file_name="./plot/wht5plt-1.pdf")
```

```
wheat_asr3 = asreml(
  response="yield",
  fixed="weed + lin(Column)",
  random="idv(Variety)",
  residual="ar1v(Row):ar1(Column)",
  options={"na_action": {"x": "include","y": "include"}},
  data=wheat)
```

```
round(wheat_asr3.summary(display=False)['loglik'], 4)
```

```
-4227.1296
```

```
wheat_asr3.variance_components()
```

```
                                          gamma         sigma  ...  bounds %ch
idv(Variety)!IDV_V                     1.143783  88998.823479  ...       P   0
ar1v(Row):ar1(Column)!Residual!SCA_V   1.000000  77810.938252  ...       P   0
ar1v(Row):ar1(Column)!Row!AR_R         0.671514      0.671514  ...       P   0
ar1v(Row):ar1(Column)!Column!AR_R      0.266144      0.266144  ...       P   0

[4 rows x 5 columns]
```

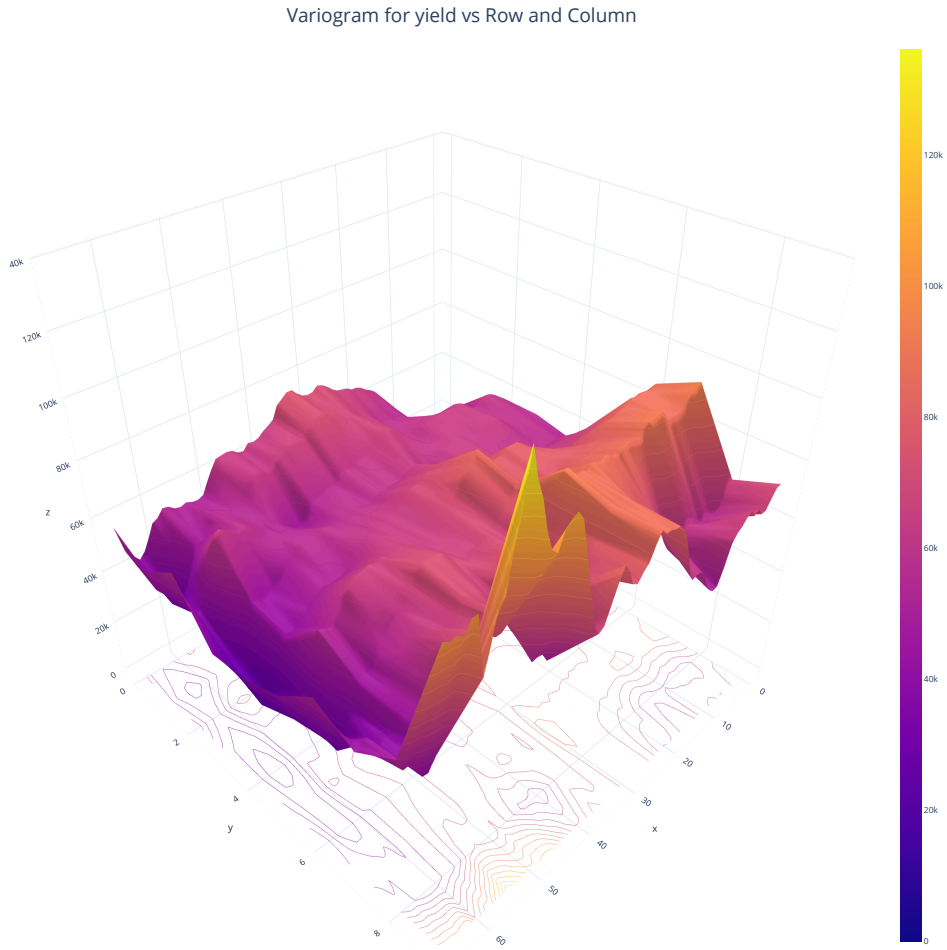

Figure 7.6: Sample variogram of the $AR1$ x $AR1$ model for the Tullibigeal data

```
wheat_asr3.wald()
```

```
              NumDF       DenDF        F-inc     P-inc
mu                1   42.515848  7086.761994  0.000000
weed              1  457.320292    91.988480  0.000000
lin(Column)       1   50.809582     8.737414  0.004718
```

```
print(wheat_asr3.coefficients(display=False).iloc[[0, 1, 2],])
```

```
                estimate  standard_error
lin(Column)   -31.186792       10.553935
weed         -182.676637       21.838227
mu           3043.508073       64.994001
```

```
variogram(wheat_asr3, file_name="./plot/wht8plt-1.pdf")
```

The linear regression of column number on yield is significant (p-value = 0.0047). The sample variogram (Figure 7.7) seems more satisfactory, though interpretation of variograms is often difficult, particularly for unreplicated trials.

The final model includes a nugget effect:

```
wheat_asr4 = asreml(
  response="yield",
  fixed="lin(Column)",
  random="idv(Variety) + idv(units)",
  sparse="weed",
  residual="ar1v(Row):ar1(Column)",
  options={"na_action": {"x": "include","y": "include"}},
  predict={"classify": "Variety:Column"},
  data=wheat)
```

```
round(wheat_asr4.summary(display=False)['loglik'], 4)
```

```
-4221.7601
```

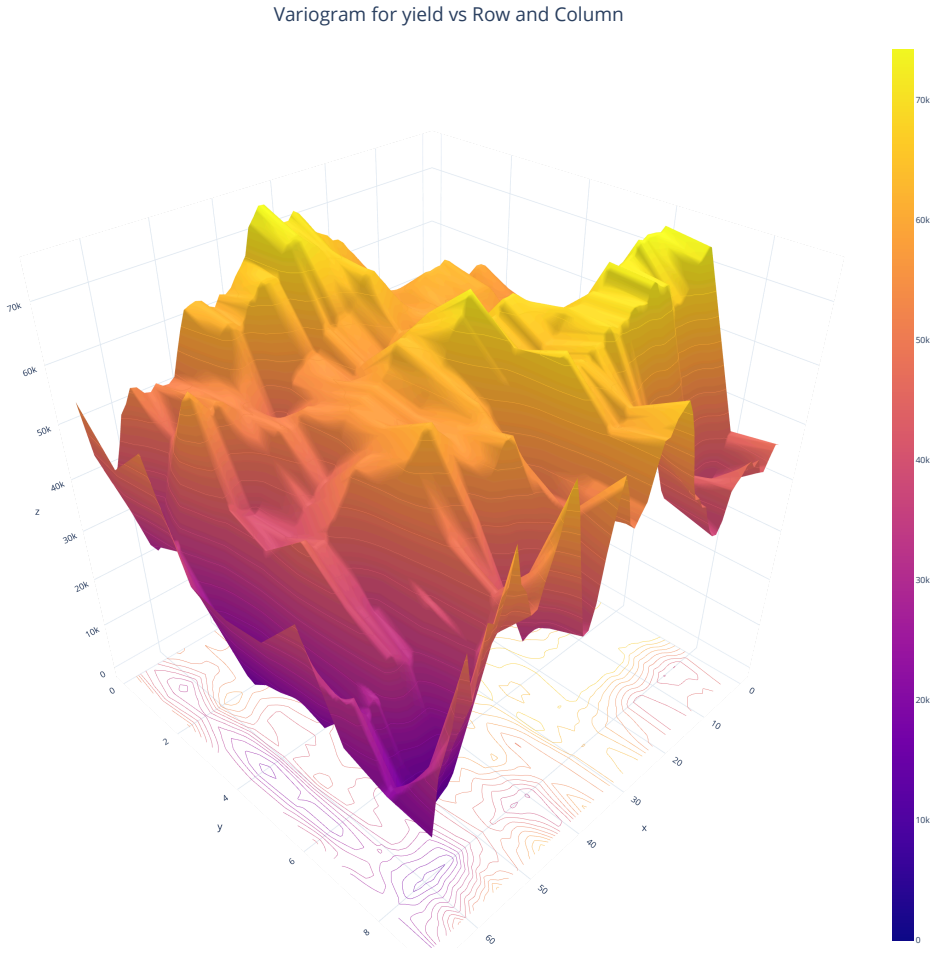

Figure 7.7: Sample variogram of the $AR1$ x $AR1$ + $lin(column)$ model for the Tullibigeal data

```
wheat_asr4.variance_components()
```

```
                                          gamma         sigma  ...  bounds %ch
idv(Variety)!IDV_V                     1.348236  73769.147253  ...       P   0
idv(units)!IDV_V                       0.556399  30443.530638  ...       P   0
ar1v(Row):ar1(Column)!Residual!SCA_V   1.000000  54715.314781  ...       P   0
ar1v(Row):ar1(Column)!Row!AR_R         0.837502      0.837502  ...       P   0
ar1v(Row):ar1(Column)!Column!AR_R      0.375382      0.375382  ...       P   0

[5 rows x 5 columns]
```

The increase in REML log-likelihood from adding the `units` term is significant. Predicted variety means can be obtained from this model using:

```
wheat_asr4.predict()
```

```
  Variety Column  Pred Values        s.e. Ecode
0       1      1  3047.629919  186.943909   E
1       1      2  3018.316161  183.589542   E
2       1      3  2989.002403  181.155809   E
3       1      4  2959.688645  179.680124   E
4       1      5  2930.374887  179.186157   E
5       1      6  2901.061129  179.682007   E
```

As a final step, and just for completeness, we will present the analyses where we separate the contribution of the check varieties from the test lines by assigning the former to be a fixed effect, and the latter as a random effect. This requires the use of the command `at()` within the model. In the following code we first create the factor `check` with a value of 1 for check varieties and 0 for test lines, and then we fit the corresponding model.

```
wheat['check'] = 0
wheat.loc[wheat['Variety'].astype(float) >= 526, 'check'] = 1
wheat = define_categorical_variables(wheat,
               variables=["Row","Column","Variety"])
```

```
wheat_asr5 = asreml(
  response="yield",
  fixed="lin(Column) + at(check, 1):Variety",
  random="at(check, 0):idv(Variety) + idv(units)",
  sparse="weed",
  residual="ar1v(Row):ar1(Column)",
  predict={"classify": "Variety"},
  options={"na_action": {"x": "include","y": "include"}},
  data=wheat)
```

Small differences are observed in the estimated variance components between this model and `wheat_asr4` with a moderate increase in the `Variety` component.

```
wheat_asr5.variance_components()
```

```
                                          gamma         sigma  ...  bounds %ch
idv(units)!IDV_V                       0.542152  29983.792842  ...       P   0
ar1v(Row):ar1(Column)!Residual!SCA_V   1.000000  55305.182613  ...       P   0
ar1v(Row):ar1(Column)!Row!AR_R         0.835953      0.835953  ...       P   0
ar1v(Row):ar1(Column)!Column!AR_R      0.378520      0.378520  ...       P   0
at(check,0):idv(Variety)!Variety!ID_V  1.345170  74394.888979  ...       P   0

[5 rows x 5 columns]
```

7.7 Paired case-control study

These data are from an experiment conducted to investigate the tolerance of rice varieties to attack by the larvae of bloodworms. The data have been kindly provided by Dr. Mark Stevens (Yanco Agricultural Institute, Australia). A full description of the experiment is given by Stevens et al. (1999). Bloodworms are a significant pest of rice in the Murray and Murrumbidgee irrigation areas and damage can result in poor establishment and substantial yield loss.

The experiment commenced with the transplanting of rice seedlings into trays. Each tray contained a total of 32 seedlings and the trays were paired so that a control tray (no bloodworms) and a treated tray (bloodworms added) were grown in a controlled environment room for the duration of the experiment. After this, rice plants were carefully extracted, the root system washed and root area determined for the tray using an image analysis system described by Stevens et al. (1999). Two pairs of trays, each pair corresponding to a different variety, were included in each run. A new batch of bloodworm larvae was used for each run. A total of 44 varieties was investigated with three replicates of each. Unfortunately the variety concurrence within runs was less than optimal. Eight varieties occurred with only one other variety, 22 with two other varieties and the remaining 14 with three different varieties.

The following sections present an exhaustive analysis of these data using equivalent univariate and multivariate techniques. It is convenient to use two data frames, one for each approach. The univariate data frame has factors `Pair`, `Run`, `Variety`, `Tmt` and variates `rootwt` and `sqrtroot`. The factor `Pair` labels pairs of trays (to which varieties are allocated) and `Tmt` is the two-level bloodworm treatment factor (control/treated). The first few lines of the univariate data frame, called `rice` are:

```
rice = pd.read_csv("./data/rice.csv", skipinitialspace=True)
print(rice.head(6))
```

```
   Pair    rootwt  Run  sqrtroot      Tmt   Variety
0     2  313.7220    1  17.71220  Control  AliCombo
1     2   86.9457    1   9.32447  Exposed  AliCombo
2    24  267.6110    1  16.35880  Control      IR36
3    24   87.1914    1   9.33763  Exposed      IR36
4     3  172.7230    2  13.14240  Control  AliCombo
5     3   26.0771    2   5.10657  Exposed  AliCombo
```

The multivariate data frame, called `riceMV`, contains factors `Variety` and `Run` and variates for root weight and square-root of root weight for both the control and exposed treatments (`yc`, `ye`, `syc`, `sye` respectively), and its first few lines are:

```
riceMV = pd.read_csv("./data/riceMV.csv", skipinitialspace=True)
print(riceMV.head(6))
```

```
   Pair  Run    Variety       yc       ye      syc      sye
0     1   60   AliCombo  185.403  70.1814  13.6163  8.37743
1     2    1   AliCombo  313.722  86.9457  17.7122  9.32447
2     3    2   AliCombo  172.723  26.0771  13.1424  5.10657
3     4    3  Bluebelle  193.056  41.9249  13.8944  6.47494
4     5    4  Bluebelle  193.119  68.3390  13.8967  8.26674
5     6    5  Bluebelle  111.228   3.5935  10.5465  1.89565
```

A plot of the treated vs the control root area (on the square root scale) for each variety is shown in Figure 7.8. There is a strong dependence between the treated and control root area, which is not surprising. The aim of the experiment was to determine the tolerance of varieties to bloodworms and identify the most tolerant varieties. The definition of tolerance should allow for the fact that varieties differ in their inherent seedling vigour (as seen in Figure 7.8). The initial approach was to regress the treated root area against the control root area and define the index of vigour as the residual from this regression. This is clearly inefficient since there is error in both variables. We seek to determine an index of tolerance from the joint analysis of treated and control root area.

Standard analysis

Preliminary analyses indicated variance heterogeneity so that subsequent analyses were conducted on the square-root scale. The allocation of bloodworm treatments within varieties and varieties within runs defines a nested blocking structure of the form:

```
Run/Variety/Tmt
  = Run + Run:Variety + Run:Variety:Tmt
  = Run + Pair + Pair:Tmt
  = Run + Run:Variety + units
```

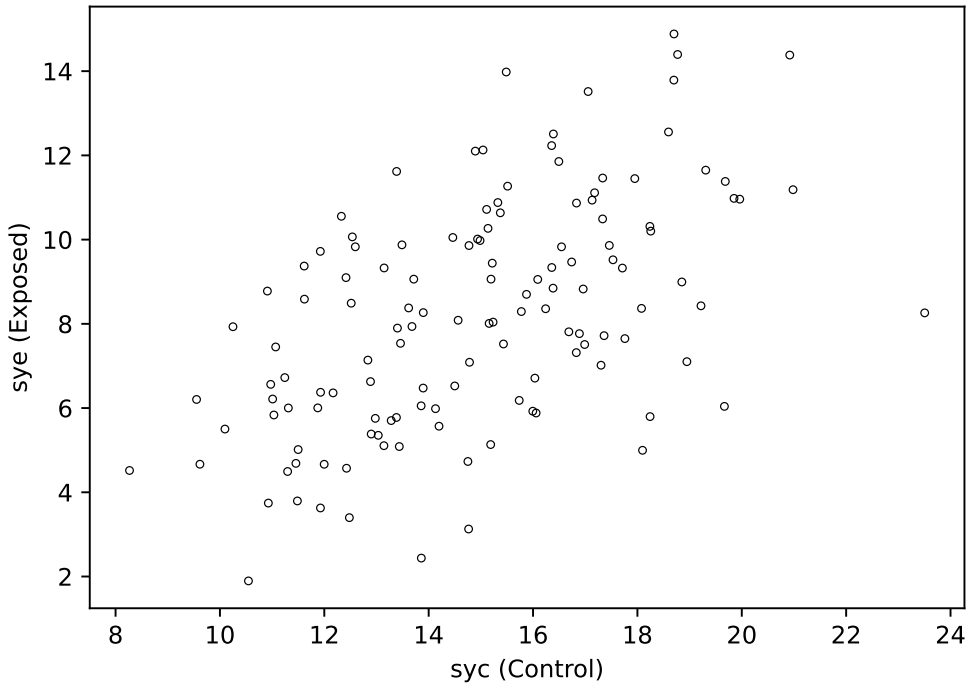

Figure 7.8: Rice bloodworm data: Plot of square root of root weight for treated versus control

There is an additional blocking term, however, due to the fact that the bloodworms within a run are derived from the same batch of larvae whereas between runs the bloodworms come from different sources. This defines a blocking structure of the form:

```
Run/Tmt/Variety
  = Run + Run:Tmt + Run:Tmt:Variety
  = Run + Run:Tmt + Pair:Tmt
```

Combining the two provides the full blocking structure for the design:

```
Run + Run:Variety + Run:Tmt + Run:Tmt:Variety
  = Run + Run:Variety + Run:Tmt + units
  = Run + Pair + Run:Tmt + Pair:Tmt
```

In line with the aims of the experiment the treatment structure comprises variety and treatment main effects and treatment by variety interactions.

In the traditional approach the terms in the block structure are regarded as random and the treatment terms as fixed. The choice of treatment terms as fixed or random depends largely on the aims of the experiment. The aim of this example is to select the *best* varieties. The definition of best is somewhat more complex since it does not involve the single trait `sqrt(rootwt)` but rather two traits, namely `sqrt(rootwt)` in the presence/absence of bloodworms. To minimize selection bias the `Variety` main effects and `Tmt:Variety` interactions are taken as random. The main effect of treatment is fitted as fixed to allow for the likely scenario that rather than a single population of treatment by variety effects there are in fact two populations (control and treated) with a different mean for each. There is evidence of this prior to analysis, with the large difference in mean `sqrt(rootwt)` for the two groups (14.93 and 8.23 for control and treated, respectively). The inclusion of `Tmt` as a fixed effect ensures that E-BLUPs of `Tmt:Variety` effects are shrunk to the correct mean (treatment means rather than an overall mean).

The model for the data is given by

$$\boldsymbol{y} = \boldsymbol{X}\boldsymbol{\tau} + \boldsymbol{Z}_1\boldsymbol{u}_1 + \boldsymbol{Z}_2\boldsymbol{u}_2 + \boldsymbol{Z}_3\boldsymbol{u}_3 + \boldsymbol{Z}_4\boldsymbol{u}_4 + \boldsymbol{Z}_5\boldsymbol{u}_5 + \boldsymbol{e} \tag{7.8}$$

where $\boldsymbol{y}$ is a vector of length $n = 264$ containing the `sqrtroot` values, $\boldsymbol{\tau}$ corresponds to a constant term and the fixed treatment contrast and $\boldsymbol{u}_1 \ldots \boldsymbol{u}_5$ correspond to random effects of: `Variety`, `Tmt:Variety`, `Run`, `Tmt:Run` and `Variety:Run`. The random effects and error are assumed to be independent Gaussian variables with zero means and variance structures: $var(\boldsymbol{u}_i) = \sigma_i^2 \boldsymbol{I}_{b_i}$ (where b_i is the length of $\boldsymbol{u}_i$, $i = 1 \ldots 5$) and $var(\boldsymbol{e}) = \sigma^2 \boldsymbol{I}_n$.

Now, we can proceed to fit the model with the code below, and we request some output.

```
rice = define_categorical_variables(rice, variables=["Pair","Run", "Tmt",
"Variety"])
```

```
rice_asr1 = asreml(
  response="sqrtroot",
  fixed="Tmt",
  random="idv(Variety) + idv(Variety):id(Tmt) + idv(Run) + idv(Pair) +
idv(Run):id(Tmt)",
  residual="units",
  data=rice
)
```

```
round(rice_asr1.summary(display=False)['loglik'], 4)
```

```
-345.2559
```

```
rice_asr1.variance_components()
```

```
                                        gamma     sigma  std.error bounds  %ch
idv(Variety)!IDV_V                   1.808248  2.377804   0.791120      P    0
idv(Run)!IDV_V                       0.244660  0.321723   0.548379      P    0
idv(Pair)!IDV_V                      0.742093  0.975835   0.388260      P    0
units!Residual!SCA_V                 1.000000  1.314977   0.297443      P    0
idv(Variety):id(Tmt)!Variety!ID_V    0.374388  0.492312   0.276419      P    0
idv(Run):id(Tmt)!Run!ID_V            1.329157  1.747811   0.479350      P    0
```

```
rice_asr1.wald()
```

```
     NumDF      DenDF        F-inc  P-inc
mu       1  53.599071  1485.652225    0.0
Tmt      1  60.417263   469.330082    0.0
```

The estimated variance components from this analysis also appear in column (a) of Table 7.7. The variance component for the `Variety` main effects is large. There is evidence of `Variety:Tmt` interactions so we may expect some discrimination between varieties in terms of tolerance to bloodworms.

Given the large difference ($p < 0.001$) between `Tmt` means we may wish to allow for heterogeneity of variance associated with `Tmt`. Thus we fit a separate `Variety:Tmt` variance for each level of `Tmt` so that instead of assuming $var(\boldsymbol{u}_2) = \sigma_2^2 \boldsymbol{I}_{88}$, we assume

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

where σ_{2c}^2 and σ_{2t}^2 are the `Variety:Tmt` interaction variances for control and treated, respectively. This model can be fitted using a diagonal variance structure for the treatment part of the interaction.

We also fit a separate `Run:Tmt` variance for each level of `Tmt` and heterogeneity at the residual level, by including an extra `at(Tmt,2):idv(units)` term. We have chosen level 2 of `Tmt` as we expect more variation for the exposed treatment and thus the extra variance component for this term should be positive.

By default, ASReml sets the parameter constraint for variance components to Positive. To allow for negative components, which may have meaning in this particular example, we must set the parameter constraints to Unconstrained. The following sequence of calls

```
rice_asr2 = asreml(
  response="sqrtroot",
  fixed="Tmt",
  random="idv(Variety) + id(Variety):diag(Tmt) + idv(Run) + idv(Pair) +
id(Run):diag(Tmt) + at(Tmt,2):idv(units)",
  residual="idv(units)",
  data=rice,
  fit_options={"process_stage": "preprocess"}
)
```

```
rice_ini = rice_asr2.initial_values(display=True)
```

```
Initial Values:
                                labels  values constraints
0                   idv(Variety)!IDV_V    0.10           P
1                       idv(Run)!IDV_V    0.10           P
2                      idv(Pair)!IDV_V    0.10           P
3            idv(units)!Residual!SCA_V    1.00           P
4  id(Variety):diag(Tmt)!Tmt_1!DIAG_V    0.02           P
5  id(Variety):diag(Tmt)!Tmt_2!DIAG_V    0.02           P
6      id(Run):diag(Tmt)!Tmt_1!DIAG_V    0.02           P
7      id(Run):diag(Tmt)!Tmt_2!DIAG_V    0.02           P
8     at(Tmt,2):idv(units)!units!ID_V    0.10           P
```

```
rice_ini["id(Variety):diag(Tmt)!Tmt_1!DIAG_V"]["constraint"] = "U"
rice_ini["id(Variety):diag(Tmt)!Tmt_2!DIAG_V"]["constraint"] = "U"
rice_ini["id(Run):diag(Tmt)!Tmt_1!DIAG_V"]["constraint"] = "U"
rice_ini["id(Run):diag(Tmt)!Tmt_2!DIAG_V"]["constraint"] = "U"
```

```
rice_asr2.fit(initial_values=rice_ini)
```

```
round(rice_asr2.summary(display = False)['loglik'], 4)
```

```
-343.2199
```

```
rice_asr2.variance_components()
```

```
                                          gamma     sigma  std.error bounds  %ch
idv(Variety)!IDV_V                     2.018545  2.334069   0.776438      P    0
idv(Run)!IDV_V                         0.275914  0.319043   0.543309      P    0
idv(Pair)!IDV_V                        0.853930  0.987410   0.381299      P    0
idv(units)!Residual!SCA_V              1.000000  1.156313   0.417376      P    0
id(Variety):diag(Tmt)!Tmt_1!DIAG_V     1.302051  1.505579   0.665211      U    0
id(Variety):diag(Tmt)!Tmt_2!DIAG_V    -0.321869 -0.372182   0.456251      U    0
id(Run):diag(Tmt)!Tmt_1!DIAG_V         1.201016  1.388750   0.635867      U    0
id(Run):diag(Tmt)!Tmt_2!DIAG_V         1.923353  2.223997   0.723812      U    0
at(Tmt,2):idv(units)!units!ID_V        0.176217  0.203762   0.631990      P    0
```

```
rice_asr2.wald()
```

```
     NumDF      DenDF        F-inc  P-inc
mu       1  56.443762  1276.717092    0.0
Tmt      1  60.597839   448.807712    0.0
```

Table 7.7: Estimated variance components from univariate analyses of bloodworm data: (a) model with homogeneous variance for all terms and (b) model with heterogeneous variance for interactions involving Tmt

source	(a) homogeneous	(b) heterogeneous control	(b) heterogeneous treated
`Variety`	2.378	2.334	
`Variety:Tmt`	0.492	1.505	-0.372
`Run`	0.321	0.319	
`Run:Tmt`	1.748	1.389	2.224
`Variety:Run` (Pair)	0.976	0.988	
`Tmt:Pair`	1.315	1.157	1.360
REML log-likelihood	-345.256	-343.526	

The estimated variance components from this analysis are given in column (b) of Table 7.7. There is no significant variance heterogeneity at the residual or `Run:Tmt` level. This indicates that the square root transformation of the data has successfully stabilized the error variance. There is, however, significant variance heterogeneity for `Variety:Tmt` interactions with the variance being much greater for the control group. This reflects the fact that in the absence of bloodworms the potential maximum root area is greater. Note that the `Variety:Tmt` interaction variance for

the treated group is negative. The negative component is meaningful (and in fact necessary and obtained by changing the constraint codes for variance parameters to `U` as described previously) in this context since it should be considered as part of the variance structure for the combined variety main effects and treatment by variety interactions. That is,

$$var(\mathbf{1}_2 \otimes \boldsymbol{u}_1 + \boldsymbol{u}_2) = \begin{bmatrix} \sigma_1^2 + \sigma_{2c}^2 & \sigma_1^2 \\ \sigma_1^2 & \sigma_1^2 + \sigma_{2t}^2 \end{bmatrix} \otimes \boldsymbol{I}_{44} \tag{7.9}$$

Using the estimates from Table 7.7 this structure is estimated as

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

Thus the variance of the variety effects in the control group (also known as the genetic variance for this group) is 3.84. The genetic variance for the treated group is much lower (1.96). The genetic correlation is $2.33/\sqrt{3.84 \times 1.96} = 0.85$ which is strong, supporting earlier indications of the dependence between the treated and control root area (Figure 7.8).

A multivariate approach

In this simple case in which the variance heterogeneity is associated with the two-level factor `Tmt`, the analysis is equivalent to a bivariate analysis in which the two traits correspond to the two levels of `Tmt`, namely `sqrtroot` for control and treated. The model for each trait is given by

$$\boldsymbol{y}_j = \boldsymbol{X}\boldsymbol{\tau}_j + \boldsymbol{Z}_v\boldsymbol{u}_{v_j} + \boldsymbol{Z}_r\boldsymbol{u}_{r_j} + \boldsymbol{e}_j \quad (j = c, t) \tag{7.10}$$

where $\boldsymbol{y}_j$ is a vector of length $n = 132$ containing the `sqrtroot` values for variate j ($j = c$ for control and $j = t$ for treated), $\boldsymbol{\tau}_j$ corresponds to a constant term by trait and $\boldsymbol{u}_{v_j}$ and $\boldsymbol{u}_{r_j}$ correspond to random variety and run effects. The design matrices are the same for both traits. The random effects and error are assumed to be independent Gaussian variables with zero means and variance structures: $var(\boldsymbol{u}_{v_j}) = \sigma_{v_j}^2\boldsymbol{I}_{44}$, $var(\boldsymbol{u}_{r_j}) = \sigma_{r_j}^2\boldsymbol{I}_{66}$ and $var(\boldsymbol{e}_j) = \sigma_j^2\boldsymbol{I}_{132}$. The bivariate model can be written as a direct extension of (7.10), namely

$$\boldsymbol{y} = (\boldsymbol{I}_2 \otimes \boldsymbol{X})\,\boldsymbol{\tau} + (\boldsymbol{I}_2 \otimes \boldsymbol{Z}_v)\,\boldsymbol{u}_v + (\boldsymbol{I}_2 \otimes \boldsymbol{Z}_r)\,\boldsymbol{u}_r + \boldsymbol{e}^* \tag{7.11}$$

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

There is an equivalence between the effects in this bivariate model and the univariate model of (7.8). The variety effects for each trait ($\boldsymbol{u}_v$ in the bivariate model) are partitioned in (7.8) into variety main effects and `tmt.variety` interactions, so that $\boldsymbol{u}_v = \mathbf{1}_2 \otimes \boldsymbol{u}_1 + \boldsymbol{u}_2$. There is a similar partitioning for the run effects and the errors (Table 7.8).

In addition to the assumptions in the models for individual traits (7.10), the bivariate analysis involves the assumptions $cov(\boldsymbol{u}_{v_c})\boldsymbol{u}_{v_t}' = \sigma_{v_{ct}}\boldsymbol{I}_{44}$, $cov(\boldsymbol{u}_{r_c})\boldsymbol{u}_{r_t}' = \sigma_{r_{ct}}\boldsymbol{I}_{66}$ and $cov(\boldsymbol{e}_c)\boldsymbol{e}_t' = \sigma_{ct}\boldsymbol{I}_{132}$. Thus, random effects and errors are correlated between traits. So, for example, the variance matrix for the variety effects for each trait is given by

Table 7.8: Equivalence of random effects in bivariate and univariate analyses

effects	bivariate (model 7.11)	univariate (model 7.8)
`trait:Variety`	$\boldsymbol{u}_v$	$\mathbf{1}_2 \otimes \boldsymbol{u}_1 + \boldsymbol{u}_2$
`trait:Run`	$\boldsymbol{u}_r$	$\mathbf{1}_2 \otimes \boldsymbol{u}_3 + \boldsymbol{u}_4$
`Pair:trait`	$\boldsymbol{e}^*$	$\mathbf{1}_2 \otimes \boldsymbol{u}_5 + \boldsymbol{e}$

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

This unstructured form for `trait:Variety` in the bivariate analysis is equivalent to the `Variety` main effect plus heterogeneous `Variety:Tmt` interaction variance structure (7.9) in the univariate analysis. Similarly, the unstructured form for `trait:Run` is equivalent to the `Run` main effect plus heterogeneous `Run:Tmt` interaction variance structure. The unstructured form for the errors (`Pair:trait`) in the bivariate analysis is equivalent to the `Pair` plus heterogeneous error (`Pair:Tmt`) variance in the univariate analysis.

Hence, the call for this bivariate model is:

```
riceMV = define_categorical_variables(riceMV, variables=["Pair","Run", "Variety"])

riceMV_asr = asreml(
    response=["syc", "sye"],
    fixed="trait",
    random="us(trait):id(Variety) + us(trait):id(Run)",
    residual="id(units):us(trait)",
    options={"wald": {"den_df": "numeric", "ss_type": "conditional"}},
    predict={"classify": "trait:Variety"},
    data=riceMV
)
```

```
round(riceMV_asr.summary(display=False)['loglik'], 4)
```

```
-343.2199
```

```
riceMV_asr.variance_components()
```

```
                                          sigma  std.error bounds  %ch
id(units):us(trait)!trait_1_1!US_V     2.143755   0.482374      P    0
id(units):us(trait)!trait_2_1!US_C     0.987419   0.381275      P    0
id(units):us(trait)!trait_2_2!US_V     2.347514   0.507687      P    0
us(trait):id(Variety)!trait_1_1!US_V   3.839612   1.107211      P    0
us(trait):id(Variety)!trait_2_1!US_C   2.334002   0.776365      P    0
us(trait):id(Variety)!trait_2_2!US_V   1.961813   0.728633      P    0
us(trait):id(Run)!trait_1_1!US_V       1.707858   0.653038      P    0
us(trait):id(Run)!trait_2_1!US_C       0.319108   0.543360      P    0
us(trait):id(Run)!trait_2_2!US_V       2.543161   0.795482      P    0
```

```
riceMV_asr.wald()
```

```
       NumDF       F-inc
trait      2  862.796975
```

The resultant REML log-likelihood is similar to that of the heterogeneous univariate analysis (column (b) of Table 7.7). The estimated variance parameters are summarized in Table 7.9.

Table 7.9: Estimated variance components from bivariate analysis of bloodworm data

source	control variance	treated variance	covariance
us(trait):Variety	3.84	1.96	2.33
us(trait):Run	1.71	2.55	0.32
Pair:us(trait)	2.14	2.35	0.99

The predicted variety means are obtained using:

```
riceMV_pv = riceMV_asr.predict(display=False)
```

and the first few rows are:

```
    Variety trait  Pred Values      s.e. Ecode
0  AliCombo   syc    14.953193  0.918093   E
1  AliCombo   sye     7.994077  0.799311   E
2    Amaroo   syc    16.161252  0.918096   E
3    Amaroo   sye     8.481321  0.799288   E
4   Balilla   syc    14.420150  0.918568   E
5   Balilla   sye     8.230817  0.799516   E
```

These predictions are on the square root scale; it is straightforward to *back-transform* the predicted means to the original scale of measurement. Approximate standard errors on the original scale can be calculated from a Taylor series approximation (*i.e.*, delta method). That is, if x is a random variable with $E(x) = \theta$, and $y = g(x)$ is some function of x, then $var(y) = (dy/dx)^2_\theta var(x)$ (see Kendall and Stuart (1969) p. 231, for an example). In this case, $g(x) = x^2$ and $g'(x) = dy/dx = 2x$. The following code calculates the transformed predictions and approximate standard errors:

```
pv = riceMV_pv.predict.copy()
pv['rootwt'] = pv['Pred Values'] ** 2
pv['approxSE'] = (4 * pv['Pred Values'] ** 2 * pv['s.e.'] ** 2) ** 0.5
print(pv.head(6))
```

```
    Variety trait  Pred Values      s.e. Ecode      rootwt   approxSE
0  AliCombo   syc    14.953193  0.918093   E    223.597971  27.456854
1  AliCombo   sye     7.994077  0.799311   E     63.905262  12.779504
2    Amaroo   syc    16.161252  0.918096   E    261.186073  29.675146
3    Amaroo   sye     8.481321  0.799288   E     71.932800  13.558038
4   Balilla   syc    14.420150  0.918568   E    207.940726  26.491764
5   Balilla   sye     8.230817  0.799516   E     67.746351  13.161343
```

Interpretation of results

Recall that the primary interest is varietal tolerance to bloodworms, and this could be defined in various ways. One option is to consider the regression implicit in the variance structure for the trait by variety effects. The variance structure can arise from a regression of treated variety effects on control effects, namely: $\boldsymbol{u}_{v_t} = \beta \boldsymbol{u}_{v_c} + \boldsymbol{\epsilon}$, where the slope is $\beta = \sigma_{v_{ct}}/\sigma^2_{v_c}$.

Tolerance can be defined in terms of the deviations from regression, $\boldsymbol{\epsilon}$. Varieties with large positive deviations have the greatest tolerance to bloodworms. Note that this is similar to the original approach except that the regression has been conducted at the genotypic rather than the phenotypic level. In Figure 7.9, the E-BLUPs for treated have been plotted against the E-BLUPs for control for each variety and the fitted regression line (slope = 0.61) has been drawn. Varieties with large positive deviations from the regression line include YRK3, Calrose, HR19 and WC1403.

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

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

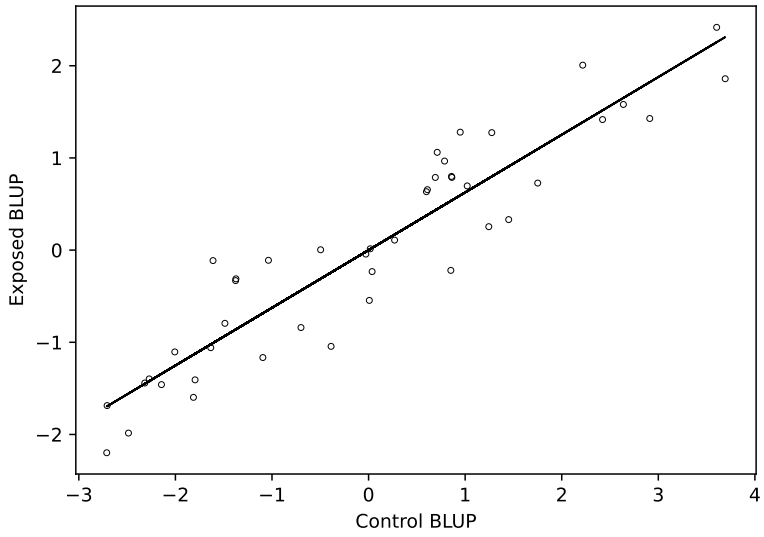

Figure 7.9: E-BLUPs for treated plotted against E-BLUPs for control

whereas

$$\begin{aligned} cov(\boldsymbol{\delta})\boldsymbol{u}'_{v_c} &= cov(\boldsymbol{u}_{v_c} - \boldsymbol{u}_{v_t})\boldsymbol{u}'_{v_c} \\ &= \left(\sigma^2_{v_c} - \sigma_{v_{ct}}\right)\boldsymbol{I}_{44} \end{aligned}$$

The independence of $\boldsymbol{\epsilon}$ and $\boldsymbol{u}_{v_c}$ and dependence between $\boldsymbol{\delta}$ and $\boldsymbol{u}_{v_c}$ is clearly illustrated in Figures 7.10 and 7.11. In this example, the two measures have provided very different rankings of the varieties. The choice of tolerance measure depends on the aim of the experiment. Here, the aim was to identify tolerance which is independent of inherent vigour so the method of deviations from regression is preferred.

7.8 Balanced longitudinal data - random coefficients and cubic smoothing splines

In this section we illustrate the use of random coefficients and cubic smoothing splines for the analysis of balanced longitudinal data. The implementation of cubic smoothing splines in ASReml is based on the mixed model formulation of Verbyla et al. (1999). The methodology has been extended so that the user can specify knot points; in the original approach, the knot points were taken to be the ordered set of unique values of the explanatory variable. The specification of knot points is particularly useful if the number of unique values in the explanatory variable is large, or if units are measured at different times.

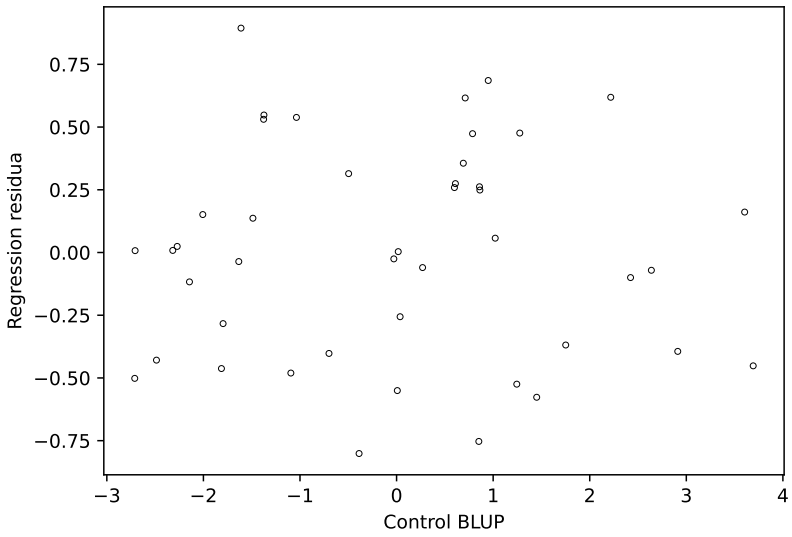

Figure 7.10: Estimated deviations from regression of treated on control for each variety plotted against estimate for control

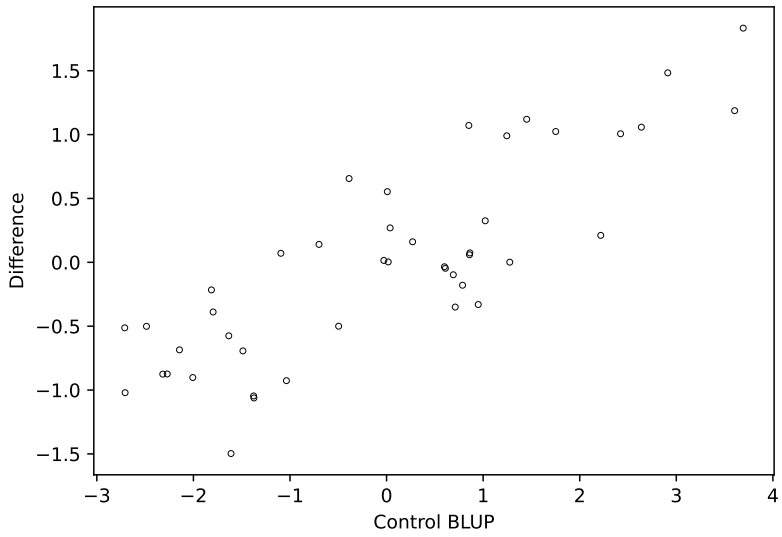

Figure 7.11: Estimated difference between control and treated for each variety plotted against estimate for control

The data set used in this illustration was originally presented by Draper and Smith (1998) and has recently been re-analysed by Pinheiro and Bates (2000). The data corresponds to trunk circumferences (in millimetres) of each of five trees taken at seven measurement times (Figure 7.12). All trees were measured at the same time so the data set is balanced. The aim of the study is unclear, though both previous analyses involved modelling the overall *growth* curve, accounting for the obvious variation in both level and shape between trees.

Pinheiro and Bates (2000) used a nonlinear mixed effects modelling approach, in which they modelled the growth curves by a three parameter logistic function of age, defined as:

$$y = \frac{\phi_1}{1 + \exp\left[-(x - \phi_2)/\phi_3\right]} \tag{7.12}$$

where y is the trunk circumference, x is the tree age (in days since December 31, 1968), ϕ_1 is the asymptotic height, ϕ_2 is the inflection point or the time at which the tree reaches $0.5\phi_1$, and ϕ_3 is the time elapsed between trees reaching half and about 3/4 of ϕ_1.

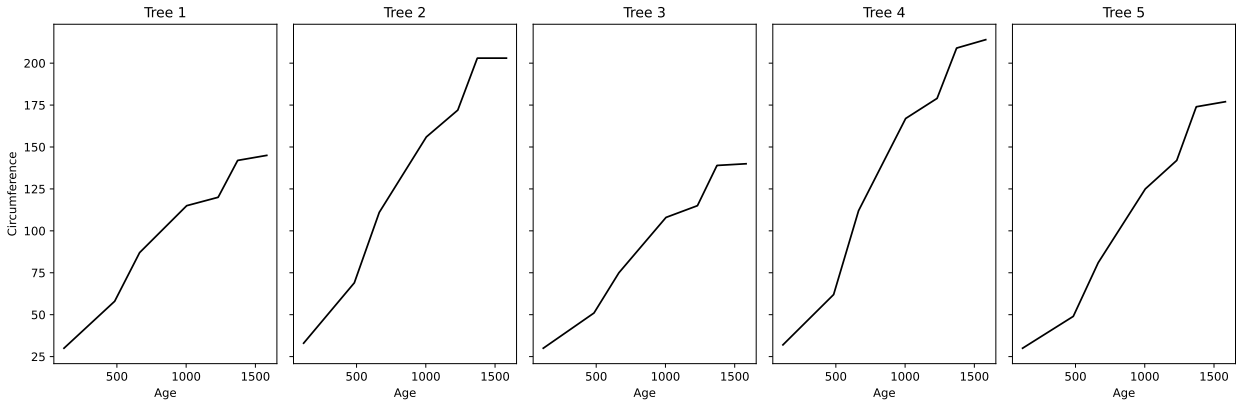

Figure 7.12: Trellis plot of trunk circumference (mm) for each tree against age (in days since December 31, 1968).

The first few lines of data frame `orange` are:

```
orange = pd.read_csv("./data/orange.csv", skipinitialspace=True)
print(orange.head(6))
```

```
   Tree     x  circ  Season
0     1   118    30  Spring
1     1   484    58  Spring
2     1   664    87  Autumn
3     1  1004   115  Autumn
4     1  1231   120  Spring
5     1  1372   142  Autumn
```

where `Tree` has five numerical categories, `x` is tree age (in days since December 31, 1968), `circ` is the trunk circumference (in mm), and `Season` has two categories (`Spring` and `Autumn`). `Season` was included after noting that tree age spans several years and converted to day of year, measurements were taken in either April/May (`Spring`) or September/October (`Autumn`). The columns `Tree` and `Season` are defined as factors as follows:

```
orange = define_categorical_variables(orange, variables=["Tree", "Season"])
```

Initially, we restrict the data set to tree 1 to demonstrate fitting cubic splines in ASReml. The model includes the intercept and linear regression of trunk circumference on x and an additional random term `spl(x)`. This random term includes a special design matrix with $7 - 2 = 5$ columns which relate to the vector, $\boldsymbol{\delta}$ whose elements $\delta_i, i = 2, \ldots, 6$ are the second differentials of the cubic spline at the knot points. The second differentials of a natural cubic spline are zero at the first and last knot points (Green and Silverman (1994)). The following code is used to fit this model:

```
orange1 = orange['data'][orange['data']['Tree'] == 1]
```

```
orange_asr = asreml(
  response="circ",
  fixed="x",
  random="spl(x)",
  residual="units",
  options={"knot_points": {"name": "x", "values": [118, 484, 664, 1004, 1231,
1372, 1582]}},
  predict={"classify": "x"},
  data=orange1
)
```

```
print(orange_asr.trace(display=False))
```

```
      LogLik          S2  DF
1 -20.904252  48.469813   5
2 -20.901270  49.152047   5
3 -20.899779  50.524229   5
```

In this example the spline knot points are specifically given in the options arguments with the key `knot_points`. These extra points have no effect in this model fit as they are exactly the seven ages present in the data file. Hence, in this instance the analysis would be the same if the `knot_points` was omitted.

The estimated variance components and Wald statistics are:

```
orange_asr.variance_components()
```

```
                          gamma      sigma  std.error bounds  %ch
spl(x)!IDV_V           0.079373   4.010272   9.680189      P    2
units!Residual!SCA_V   1.000000  50.524229  38.216203      P    0
```

```
orange_asr.wald()
```

```
    NumDF     DenDF         F-inc     P-inc
mu      1  3.495701  1373.623856  0.000012
x       1  3.495701   216.158344  0.000288
```

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

```
orange_pv = orange_asr.predict(display=False)
```

And the first and last few rows are:

```
print(orange_pv.predict.head(6))
```

```
                       x  Pred Values      s.e. Ecode
0                    118    30.831936  6.045525   E
1  0.191200000000100E+03    37.594258  5.312303   E
2  0.264400000000200E+03    44.350868  4.747851   E
3  0.337600000000300E+03    51.096053  4.365819   E
4  0.410800000000400E+03    57.824099  4.130788   E
5                    484    64.529294  3.979326   E
```

```
print(orange_pv.predict.tail(6))
```

```
                       x  Pred Values      s.e. Ecode
16                  1231   125.894000  3.536202   E
17 0.130420000000010E+04   131.097475  3.601695   E
18                  1372   135.833727  3.791171   E
19 0.144520000000010E+04   140.857367  4.151138   E
20 0.151840000000020E+04   145.815589  4.716757   E
21                  1582   150.097617  5.383339   E
```

The REML estimate of the smoothing constant and the fitted cubic smoothing spline (Figure 7.13) indicate there is some nonlinearity. The four points below the line are the spring measurements.

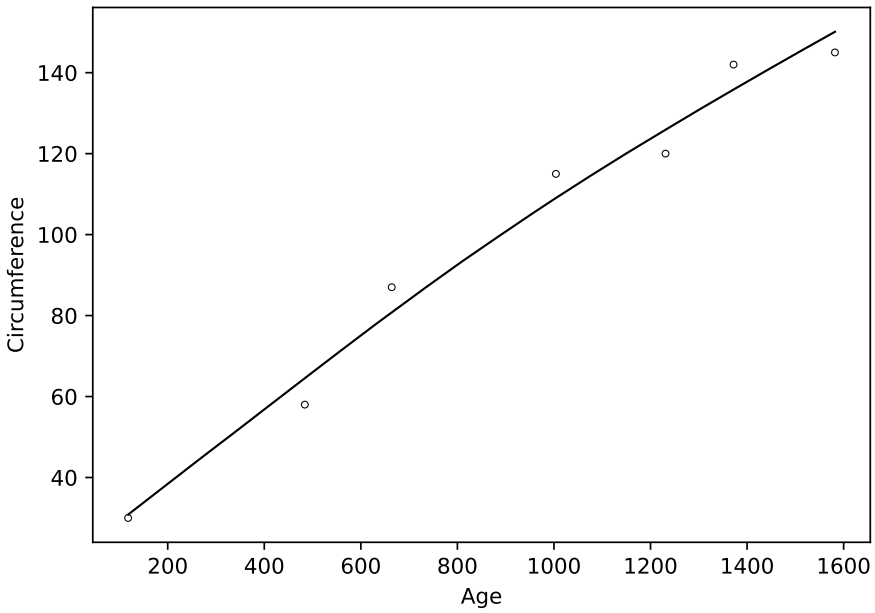

Figure 7.13: Fitted cubic smoothing spline for tree 1

Table 7.10: ANOVA decomposition for the orange data

stratum	decomposition	type	df or ne
(Intercept)	1	f	1
`Age`			
	`x`	f	1
	`spl(x)`	r	5
	`residual`	r	7
`Tree`			
	`Tree`	rc	5
`Age:Tree`			
	`x:Tree`	rc	5
	`spl(x):Tree`	r	25
`error`		r	

An analysis of variance decomposition for the full data set is given in Table 7.10, following Verbyla et al. (1999). An overall spline is included as well as tree deviation splines. We note that the intercept and slope for the tree deviation splines are assumed to be random effects, which is consistent with Verbyla et al. (1999). In this sense, the tree deviation splines play a role in modelling the conditional curves for each tree and variance modelling. The intercept and slope for each tree are included as

random coefficients (denoted by **rc** in Table 7.10). Thus, if $\boldsymbol{U}^{5\times 2}$ is the matrix of intercepts (column 1) and slopes (column 2) for each tree, then we assume that

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

where $\boldsymbol{\Sigma}$ is a 2×2 symmetric positive definite matrix. Non-smooth variation can be modelled at the overall mean (across trees) level and this is achieved by including the factor `dev(x)` as a random term. This full model is:

```
orange1_asr = asreml(
  response="circ",
  fixed="x",
  random="str(Tree + x:Tree, diag(2):id(5)) + spl(x) + spl(x):id(Tree) +
idv(dev(x))",
  residual="units",
  options={"knot_points": {"name": "x", "values": [118, 484, 664, 1004, 1231,
1372, 1582]}},
  data=orange
)
```

Note that the command `str()` will apply the direct product variance defined by $\boldsymbol{\Sigma}_2 \otimes \boldsymbol{I}_5$ to the combined set of terms `Tree` and `x:Tree`. In this case Σ_2 is a diagonal matrix of dimension 2 but other structures can be defined.

Table 7.11 presents the sequence of fitted models. We stress the importance of model building in these settings, where we generally start with relatively simple variance models and update to more complex variance models if appropriate. Note that the REML log-likelihood values for Models 1 and 2 are comparable and likewise for Models 3 to 6. The REML log-likelihoods are not comparable between these groups because of the inclusion of the fixed `Season` factor.

We begin by modelling the variance matrix $\boldsymbol{\Sigma}$ for the intercept and slope for each tree as a diagonal matrix as there is no point including a covariance component between the intercept and slope if the variance component(s) for one (or both) is zero. Model 1 also does not include a non-smooth component at the overall level (that is, `dev(x)`).

The call and variance components for model 1 are:

```
orange1_asr = asreml(
  response="circ",
  fixed="x",
  random="str(Tree + x:Tree, diag(2):id(5)) + spl(x) + spl(x):id(Tree)",
  residual="units",
  options={"knot_points": {"name": "x", "values": [118, 484, 664, 1004, 1231,
1372, 1582]}},
  predict={"classify": "x:Tree"},
  data=orange
)
```

Table 7.11: Sequence of models fitted to the `orange` data

	model					
term	1	2	3	4	5	6
`Tree`	y	y	y	y	y	y
`x:Tree`	y	y	y	y	y	y
`cov(Tree, x:Tree)`	n	n	n	n	n	y
`spl(x)`	y	y	y	y	n	y
`spl(x):Tree`	y	y	y	n	y	y
`dev(x)`	n	y	y	n	n	n
`Season`	n	n	y	y	y	y
REML log-likelihood	-97.78	-94.07	-87.95	-91.22	-90.18	-87.43

```
orange1_asr.variance_components()
```

```
                            gamma       sigma   std.error bounds  %ch
spl(x)!IDV_V             91.811300  698.469011  398.256188      P   20
spl(x):id(Tree)!IDV_V     0.923743    7.027523    3.893726      P   34
units!Residual!SCA_V      1.000000    7.607658    4.199535      P    0
diag(2):id(5)!2_1!DIAG_V  4.326679   32.915898   22.991458      P   20
diag(2):id(5)!2_2!DIAG_V  0.000086    0.000657    0.000430      P   13
```

The fitted curves from this model are shown in Figure 7.14. The fit is unacceptable because the spline has picked up too much curvature, suggesting there may be systematic non-smooth variation at the overall level. This can be formally examined by including the `dev(x)` term as a random effect.

```
orange6_asr = asreml(
  response="circ",
  fixed="x + Season",
  random="str(Tree + x:Tree, us(2, init = [5.0, -0.01, 0.0001]):id(5)) + spl(x) +
spl(x):id(Tree)",
  residual="units",
  options={"knot_points": {"name": "x", "values": [118, 484, 664, 1004, 1231,
1372, 1582]}},
  predict={"classify": "x:Tree"},
  data=orange
)
```

Model 2 increased the REML log-likelihood by 3.70 ($P < 0.05$) with the `spl(x)` smoothing constant approaching the boundary. The `Season` factor provides a possible explanation. When included in

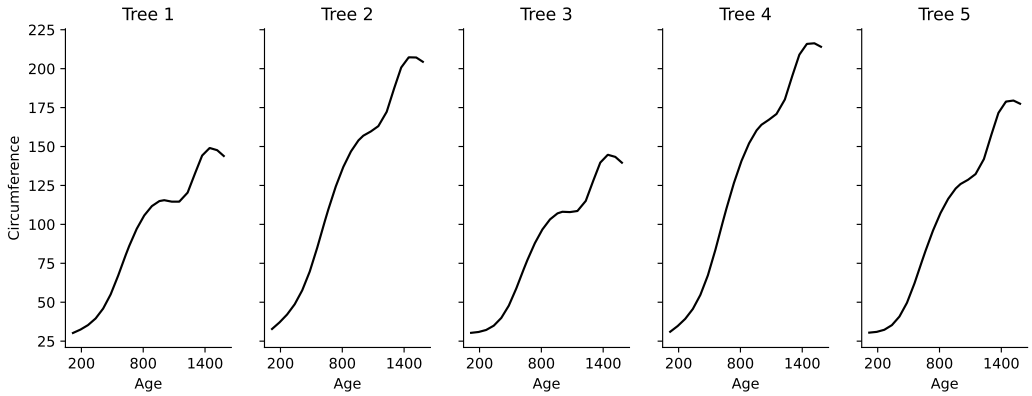

Figure 7.14: Plot of fitted cubic smoothing spline for model 1

Model 3, `Season` has a Wald statistic of 107.3 ($P < 0.01$) and `dev(x)` becomes bounded. The spring measurements are lower than the autumn measurements so growth is slower in winter. Models 4 and 5 successively examined each term, indicating that both smoothing constants are significant. Model 6 includes the covariance parameter between the intercept and slope for each tree; this ensures that the model will be translation invariant. This model requires care in the choice of starting values. The call illustrating an alternative method for specifying initial values, and the estimated variance components for model 6 are:

```
orange6_asr.variance_components()
```

```
                          gamma      sigma  std.error bounds  %ch
spl(x)!IDV_V           2.165616  12.308922  11.201254      P    0
spl(x):id(Tree)!IDV_V  1.373304   7.805582   5.210782      P    1
units!Residual!SCA_V   1.000000   5.683798   3.297043      P    0
us(2):id(5)!2_1_1!US_V 5.597507  31.815097  25.128621      P    0
us(2):id(5)!2_2_1!US_C -0.012369 -0.070304   0.082348      P    0
us(2):id(5)!2_2_2!US_V 0.000108   0.000614   0.000434      P    0
```

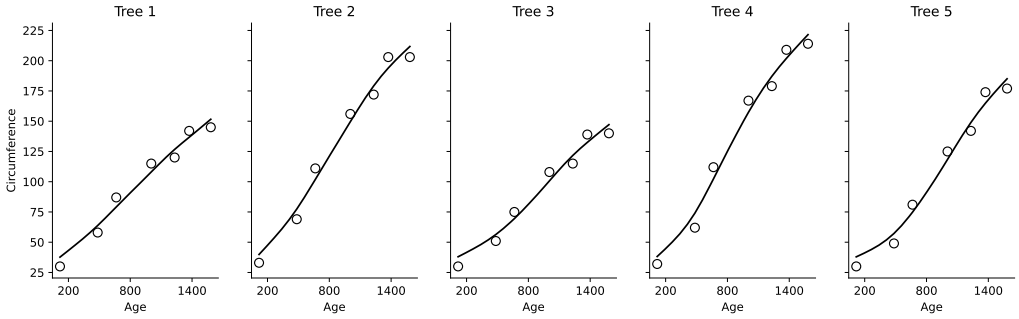

Figure 7.15: Fitted values adjusted for ‘Season‘

The fitted values for individual trees (adjusted for `Season`) from Model 6 together with a marginal prediction and approximate confidence intervals (2× standard error of prediction) are shown in Figure 7.15. The conclusions from this analysis are quite different from those obtained by the nonlinear mixed effects analysis. The individual curves for each tree are not convincingly modelled by a logistic function. There is a distinct pattern in the residuals shown in Pinheiro and Bates (2000, 340), which is consistent for all trees; this is modelled here by the `Season` term.

Appendix A

Some technical details about model fitting in ASReml

A.1 Sparse *versus* dense

The terms in the linear mixed model are partitioned into two sets; a dense set and a sparse set. The partition is defined by the fixed formula; all terms in the `fixed` formula are included in the dense set, while all terms in the random and sparse formulae are included in the sparse set. The inverse coefficient matrix is fully formed for the terms in the dense set which are fitted using dense equations. The inverse coefficient matrix is only partially formed for terms in the sparse set. Typically, the sparse set is large, resulting in savings in memory and computing. A consequence is that the variance matrix of the BLUEs and BLUPs is only available for terms in the dense portion.

As indicated, random effects are, by default, part of the sparse set. However, genetic relationship matrices, for example, are typically quite large (in the order of several thousand) and dense, and it can be more efficient to process such terms as dense. The `gdense` parameter set to `True` in the `options` argument can be used to include terms in the `random` formula in the dense set of equations for processing. In the following example, we present the ways to include the equations in the dense set for processing.

```
asreml(
    response="P",
    fixed="1",
    random="vm(ID, ainv)",
    data={"data": phenotype, "ainv": ainv},
    options={"gdense": True})
```

A.2 Aliasing and singularities

A singularity occurs when there is either:

- a linear dependence in the model and therefore no information left to estimate the corresponding effect,
- no data for a given fixed effect, or
- no data for a simple (uncorrelated) random effect.

The REML routines handle singularities by deleting the equations in question. Since the equations are solved from the bottom up, the first level (and hence the last level processed) of a factor is the one that will be declared singular and dropped from the model.

Important Singularities in the *sparse* terms of the model may change with changes in the random terms included in the model. If this happens it will mean that changes in the REML log-likelihood are not valid for testing the changes made to the random model. A likelihood ratio test, under REML, is not valid if the fixed model has changed.

A.2.1 Examples of aliasing

The sequence of examples in Table A.1 are presented to facilitate an understanding of over-parameterized models. It is assumed that `Var` is defined with four levels, `Trt` with three levels and `Rep` with three levels, and that all `Var:Trt` combinations are present in the data.

Table A.1: Examples of aliasing

model	singularities	description
`fixed = "-1 + Var",` `random = "Rep"`	0	`Var` fully fitted
`fixed = "Var",` `random = "Rep"`	1	first level of `Var` dropped
`fixed = "-1 + Var + Trt",` `random = "Rep"`	1	`Var` fully specified, first level of `Trt` dropped from the models
`fixed = "Var + Trt + Var:Trt",` `random = "Rep"`	8	first level of both `Var` and `Trt` dropped from the model, together with subsequent interactions

`fixed = "Var + Trt",`
`random = "Rep",`
`sparse = "Var:Trt"`
	8	`Var:Trt` fully specified; `(Intercept)`, `Var` and `Trt` completely singular and dropped from the model

Appendix B

Variance structures in ASReml

B.1 Variance and correlation structure functions

Table B.1 presents the full range of variance models available in ASReml with their algebraic descriptions and numbers of parameters. In most cases the algebraic form is for the correlation model (`id()` to `agau()`). However, the models from `diag()` onwards are additional heterogeneous variance models. Recall from Section 4.2 the algebraic forms of the homogeneous and heterogeneous variance models are determined as follows.

Let $\boldsymbol{C}^{(\omega\times\omega)} = [C_{ij}]$ be the correlation matrix for a particular correlation model. If $\boldsymbol{\Sigma}^{(\omega\times\omega)}$ is the corresponding homogeneous variance matrix then

$$\boldsymbol{\Sigma} \;=\; \sigma^2\boldsymbol{C}$$

and has just one more parameter than the correlation model. For example, the homogeneous variance model corresponding to the `id()` correlation model has variance matrix $\boldsymbol{\Sigma} = \sigma^2 I_\omega$ (specified `idv()` in the ASReml function call, see below) and one parameter. Likewise, if $\boldsymbol{\Sigma}_h^{(\omega\times\omega)}$ is the heterogeneous variance matrix corresponding to $\boldsymbol{C}$, then

$$\boldsymbol{\Sigma}_h \;=\; \boldsymbol{DCD}$$

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

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

(specified `idh()` in the `asreml()` call, see following table) and involves the ω parameters $\sigma_1^2 \ldots \sigma_\omega^2$.

Table B.1: Details of the available variance models

function name	description	algebraic form	corr	number of parameters: homog. variance	number of parameters: heterog. variance
Correlation models					
`id()`	identity	$C_{ii} = 1,\ C_{ij} = 0,\ \ i \neq j$	0	1	ω
`ar1()`	1^{st} order autoregressive	$C_{ii} = 1,\ C_{i+1,i} = \phi_1$ $C_{ij} = \phi_1 C_{i-1,j},\ i > j+1$ $\lvert\phi_1\rvert < 1$	1	2	$1 + \omega$
`ar2()`	2^{nd} order autoregressive	$C_{ii} = 1,$ $C_{i+1,i} = \phi_1/(1-\phi_2)$ $C_{ij} = \phi_1 C_{i-1,j} + \phi_2 C_{i-2,j},\ i > j+1$ $\lvert\phi_1 \pm \phi_2\rvert < 1,$ $\lvert\phi_1\rvert < 1,\ \lvert\phi_2\rvert < 1$	2	3	$2 + \omega$
`ar3()`	3^{rd} order autoregressive	$C_{ii} = 1, \Omega = 1 - \phi_2 - \phi_3(\phi_1 + \phi_3),$ $C_{i+1,i} = (\phi_1 + \phi_2\phi_3)/\Omega,$ $C_{i+2,i} = (\phi_1(\phi_1 + \phi_3) + \phi_2(1-\phi_2))/\Omega,$ $C_{ij} = \phi_1 C_{i-1,j} + \phi_2 C_{i-2,j} + \phi_3 C_{i-3,j},\ i > j+2$ $\lvert\phi_1\rvert < (1-\phi_2), \lvert\phi_2\rvert < 1, \lvert\phi_3\rvert < 1$	3	4	$3 + \omega$
`sar()`	symmetric autoregressive	$C_{ii} = 1,$ $C_{i+1,i} = \phi_1/(1 + \phi_1^2/4)$ $C_{ij} = \phi_1 C_{i-1,j} - \phi_1^2/4\, C_{i-2,j},$ $i > j+1$ $\lvert\phi_1\rvert < 1$	1	2	$1 + \omega$
`sar2()`	constrained autoregressive 3 used for competition	as for AR3 using $\phi_1 = \gamma_1 + 2\gamma_2,$ $\phi_2 = -\gamma_2(2\gamma_1 + \gamma_2),$ $\phi_3 = \gamma_1\gamma_2^2$	2	3	$2 + \omega$
`ma1()`	1^{st} order moving average	$C_{ii} = 1,$ $C_{i+1,i} = -\theta_1/(1 + \theta_1^2)$ $C_{ji} = 0,\ j > i+1$ $\lvert\theta_1\rvert < 1$	1	2	$1 + \omega$

Details of the available variance models

function name	description	algebraic form	number of parameters		
			corr	homog. variance	heterog. variance
`ma2()`	2^{nd} order moving average	$C_{ii} = 1$, $C_{i+1,i} = -\theta_1(1-\theta_2)/(1+\theta_1^2+\theta_2^2)$ $C_{i+2,i} = -\theta_2/(1+\theta_1^2+\theta_2^2)$ $C_{ji} = 0,\ j > i+2$ $\theta_2 \pm \theta_1 < 1$ $\lvert\theta_1\rvert < 1,\ \lvert\theta_2\rvert < 1$	2	3	$2+\omega$
`arma()`	autoregressive moving average	$C_{ii} = 1$, $C_{i+1,i} = (\theta-\phi)(1-\theta\phi)/(1+$ $\theta^2 - 2\theta\phi)$ $C_{ji} = \phi C_{j-1,i},\ j > i+1$ $\lvert\theta\rvert < 1,\ \lvert\phi\rvert < 1$	2	3	$2+\omega$
`cor()`	uniform correlation	$C_{ii} = 1,\ C_{ij} = \theta,\ i \neq j$	1	2	$1+\omega$
`corb()`	banded correlation	$C_{ii} = 1$ $C_{i+j,i} = \phi_j,\ 1 \leq j < \omega$ $\lvert\phi_j\rvert < 1$	j	$j+1$	$j+\omega$
`corg()`	general correlation `corgh()` = `us()`	$C_{ii} = 1$ $C_{ij} = \phi_{ij},\ i \neq j$ $\lvert\phi_{ij}\rvert < 1$	$\frac{\omega(\omega-1)}{2}$	$\frac{\omega(\omega-1)}{2}+1$	$\frac{\omega(\omega-1)}{2}+\omega$
One-dimensional unequally spaced power models					
`exp()`	exponential	$C_{ii} = 1$ $C_{ij} = \phi^{\lvert x_i - x_j\rvert},\ i \neq j$ x_i are *coordinates* $\lvert\phi\rvert < 1$	1	2	$1+\omega$
`gau()`	gaussian	$C_{ii} = 1$ $C_{ij} = \phi^{(x_i-x_j)^2}$ x_i are *coordinates* $\lvert\phi\rvert < 1$	1	2	$1+\omega$
`lvr()`	linear variance	$C_{ij} = (1-\theta_{ij})$ $0 < \phi_1$	1	2	$1+\omega$

Details of the available variance models

function name	description	algebraic form	corr	number of parameters: homog. variance	number of parameters: heterog. variance
Two-dimensional irregularly spaced power models					
`iexp()`	isotropic exponential	$C_{ii} = 1$ $C_{ij} = \phi^{\lvert x_i - x_j\rvert + \lvert y_i - y_j\rvert}$ $\boldsymbol{x}$ and (y) vectors of coordinates $\lvert\phi\rvert < 1$	1	2	$1 + \omega$
`igau()`	isotropic gaussian	$C_{ii} = 1$ $C_{ij} = \phi^{(x_i - x_j)^2 + (y_i - y_j)^2}$ $\boldsymbol{x}$ and (y) vectors of coordinates $\lvert\phi\rvert < 1$	1	2	$1 + \omega$
`ieuc()`	isotropic euclidean	$C_{ii} = 1$ $C_{ij} = \phi^{\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}}$ $\boldsymbol{x}$ and (y) vectors of coordinates $\lvert\phi\rvert < 1$	1	2	$1 + \omega$
`sph()`	spherical	$C_{ij} = 1 - \frac{3}{2}\theta_{ij} + \frac{1}{2}\theta_{ij}^3$ $0 < \phi_1$	1	2	$1 + \omega$
`cir()`	circular[†]	$C_{ij} = 1$ $-\frac{2}{\pi}(\theta_{ij}\sqrt{1 - \theta_{ij}^2} + \sin^{-1}\theta_{ij})$ $0 < \phi_1$	1	2	$1 + \omega$
`aexp()`	anisotropic exponential	$C_{ii} = 1$ $C_{ij} = \phi_1^{\lvert x_i - x_j\rvert}\phi_2^{\lvert y_i - y_j\rvert}$ $\boldsymbol{x}$ and (y) vectors of coordinates $\lvert\phi_1\rvert < 1$, $\lvert\phi_2\rvert < 1$	2	3	$2 + \omega$
`agau()`	anisotropic gaussian	$C_{ii} = 1$ $C_{ij} = \phi_1^{(x_i - x_j)^2}\phi_2^{(y_i - y_j)^2}$ $\boldsymbol{x}$ and (y) vectors of coordinates $\lvert\phi_1\rvert < 1$, $\lvert\phi_2\rvert < 1$	2	3	$2 + \omega$
`mtrn()`	Matérn with first $1 \le k \le 5$ parameters specified by the user	C_{ij} =Matérn: $\phi > 0$ range, ν shape(0.5) $\delta > 0$ anisotropy ratio(1), α anisotropy angle(0), $\lambda(1\lvert 2)$ metric(2)	k	$k + 1$	$k + \omega$

Details of the available variance models

function name	description	algebraic form	number of parameters		
			corr	homog. variance	heterog. variance
Heterogeneous variance models					
`diag()`	diagonal = `idh()`	$\boldsymbol{\Sigma}_{ii} = \phi_i \ \boldsymbol{\Sigma}_{ij} = 0, \ i \neq j$	-	-	ω
`us()`	unstructured general covariance	$\boldsymbol{\Sigma}_{ij} = \phi_{ij}$	-	-	$\frac{\omega(\omega+1)}{2}$
`ante(, k)`	antedependence order `k` $1 \leq order \leq \omega - 1$	$\boldsymbol{\Sigma}^{-1} = \boldsymbol{UDU'}$ $D_{ii} = d_i, \ D_{ij} = 0, \ i \neq j$ $U_{ii} = 1, \ U_{ij} = u_{ij}, \ 1 \leq j - i \leq$ order $U_{ij} = 0, \ i > j$	-	-	$(k+1)(\omega - k/2)$
`chol(, k)`	cholesky order `k` $1 \leq order \leq \omega - 1$	$\boldsymbol{\Sigma} = \boldsymbol{LDL'}$ $D_{ii} = d_i, \ D_{ij} = 0, \ i \neq j$ $L_{ii} = 1, \ L_{ij} = l_{ij}, \ 1 \leq i-j \leq$ order	-	-	$(k+1)(\omega - k/2)$
`sfa(, k)`	factor analytic order `k`, scaled form	$\boldsymbol{\Sigma} = \boldsymbol{DCD}$, $\boldsymbol{C} = \boldsymbol{FF'} + \boldsymbol{E}$, $\boldsymbol{F}$ *contains correlation factors* $\boldsymbol{E}$ *diagonal* $\boldsymbol{DD} = \text{diag}\,(\boldsymbol{\Sigma})$	-	-	$k\omega + \omega$
`fa(, k)`	factor analytic order `k`, sparse form	$\boldsymbol{\Sigma} = \boldsymbol{\Gamma\Gamma'} + \boldsymbol{\Psi}$, $\boldsymbol{\Gamma}$ *contains covariance factors* $\boldsymbol{\Psi}$ *contains specific variance*	-	-	$k\omega + \omega$
`facv(, k)`	factor analytic order `k`, covariance form	$\boldsymbol{\Sigma} = \boldsymbol{\Gamma\Gamma'} + \boldsymbol{\Psi}$, $\boldsymbol{\Gamma}$ *contains covariance factors* $\boldsymbol{\Psi}$ *contains specific variance*	-	-	$k\omega + \omega$
`rr(, k)`	reduced rank order `k`	$\boldsymbol{\Sigma} = \boldsymbol{\Gamma\Gamma'}$ $\boldsymbol{\Gamma}$ *contains covariance factors*	-	-	$k\omega$
Known variance structures					
`vm()`	known variance model (genetic relationship matrix or its inverse)		-	-	-
General variance models					
`str()`	variance model relating to a sequence of terms in the model		-	-	-
`dsum()`	direct sum structures for the residual error term		-	-	-

Bibliography

Breslow, N. E., and D. G. Clayton. 1993. Approximate inference in generalized linear mixed models. *Journal of the American Statistical Association*. 88:9–25.

Cox, D. R., and D. V. Hinkley. 1974. *Theoretical statistics*. Chapman; Hall.

Cox, D. R., and E. J. Snell. 1981. *Applied statistics; principles and examples*. Chapman; Hall.

Cressie, N. A. C. 1991. *Statistics for spatial data*. John Wiley; Sons.

Cullis, B. R., and A. C. Gleeson. 1991. Spatial analysis of field experiments - an extension to two dimensions. *Biometrics*. 47:1449–1460.

Cullis, B. R., A. C. Gleeson, W. J. Lill, J. A. Fisher, and B. J. Read. 1989. A new procedure for the analysis of early generation variety trials. *Applied Statistics*. 38:361–375.

Cullis, B., B. Gogel, A. Verbyla, and R. Thompson. 1998. Spatial analysis of multi-environment early generation variety trials. *Biometrics*. :1–18.

Dempster, A. P., M. R. Selwyn, C. M. Patel, and A. J. Roth. 1984. Statistical and computational aspects of mixed model analysis. *Applied Statistics*. 33:203–214.

Draper, N. R., and H. Smith. 1998. *Applied regression analysis*. 3rd ed. John Wyley; Sons, New York.

Gelfand, A. E., M. Fuentes, P. Guttorp, and P. Diggle. 2010. *Handbook of spatial statistics*. Taylor & Francis.

Gilmour, A. R., B. R. Cullis, and A. P. Verbyla. 1997. Accounting for natural and extraneous variation in the analysis of field experiments. *Journal of Agricultural, Biological, and Environmental Statistics*. 2:269–273.

Gilmour, A. R., B. R. Cullis, S. J. Welham, B. J. Gogel, and R. Thompson. 2004. An efficient computing strategy for prediction in mixed linear models. *Computational Statistics and Data Analysis*. 44:571–586.

Gilmour, A. R., B. J. Gogel, B. R. Cullis, S. J. Welham, G. S. A., and R. Thompson. 2026.

ASReml user guide release 4.3. VSN International, 5 The Waterhouse, Waterhouse St, Hemel Hempstead, HP1 1ES, UK.

Gilmour, A. R., R. Thompson, and B. R. Cullis. 1995. Average information REML: An efficient algorithm for variance parameter estimation in linear mixed models. *Biometrics*. 51:1440–1450.

Gleeson, A. C., and B. R. Cullis. 1987. Residual maximum likelihood (REML) estimation of a neighbour model for field experiments. *Biometrics*. 43:277–288.

Gogel, B. J. 1997. Spatial analysis of multi-environment variety trials/beverley j. gogel. PhD thesis.

Green, P. J., and B. W. Silverman. 1994. *Nonparametric regression and generalized linear models*. Chapman; Hall.

Harvey, W. R. 1977. *Users' guide to LSML76*. Ohio State University, Columbus.

Kendall, M. G., and A. Stuart. 1969. *The advanced theory of statistics*. Charles Griffin, London.

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

Lane, P. W., and J. A. Nelder. 1982. Analysis of covariance and standardisation as instances of predicton. *Biometrics*. 38:613–621.

McCullagh, P., and J. A. Nelder. 1994. *Generalized linear models*. 2nd ed. Chapman; Hall, London.

Meuwissen, T. H. E., and Z. Luo. 1992. Computing inbreeding coefficients in large populations. *Genetics Selection Evolution*. 24.

Nelder, J. A. 1977. A reformulation of linear models. *Journal of the Royal Statistical Society, Series A*. 140:48–76.

Nelder, J. A. 1994. The statistics of linear models: Back to basics. *Statistics and Computing*. 4:221–234.

Patterson, H. D., and R. Thompson. 1971. Recovery of interblock information when block sizes are unequal. *Biometrika*. 31:100–109.

Pinheiro, J. C., and D. M. Bates. 2000. *Mixed-effects models in S and S-PLUS*. Springer-Verlaag.

Quaas, R. L., and E. J. Pollak. 1980. Mixed model methodology for farm and ranch beef cattle testing programs. *Journal of Animal Science*. 51(6):1277–1287.

Robinson, G. K. 1991. That BLUP is a good thing: The estimation of random effects. *Statistical Science*. 6:15–51.

Searle, S. R. 1971. *Linear models*. John Wiley; Sons, inc.

Searle, S. R., G. Casella, and C. E. McCulloch. 1992. *Variance components*. J. W. Wiley, New York.

Self, S. C., and K. Y. Liang. 1987. Asymptotic properties of maximum likelihood estimators and likelihood ratio tests under non-standard conditions. *Journal of the American Statistical Society*. 82:605–610.

Stevens, M. M., K. M. Fox, G. N. Warren, B. R. Cullis, N. E. Coombes, and L. G. Lewin. 1999. An image analysis technique for assessing resistance in rice cultivars to root-feeding chironomid midge larvae (Diptera: Chironomidae). *Field Crops Research*. 66:25–26.

Thompson, R., B. R. Cullis, A. Smith, and A. R. Gilmour. 2003. A sparse implementation of the average information algorithm for factor analytic and reduced rank variance models. *Australian and New Zealand Journal of Statistics*. 45:445–459.

Verbyla, A. P. 1990. A conditional derivation of residual maximum likelihood. *Australian Journal of Statistics*. 32(2):227–230.

Verbyla, A. P., B. R. Cullis, M. G. Kenward, and S. J. Welham. 1999. The analysis of designed experiments and longitudinal data using smoothing splines. *Journal of the Royal Statistical Society, Series C*. :269–311.

Welham, S. J., B. R. Cullis, B. J. Gogel, A. R. Gilmour, and R. Thompson. 2004. Prediction in linear mixed models. *Australian and New Zealand Journal of Statistics*. 46:325–347.

Welham, S. J., and R. Thompson. 1997. Likelihood ratio tests for fixed model terms using residual maximum likelihood. *Journal of the Royal Statistical Society, Series B*. 59:701–714.

Wolfinger, R. D. 1996. Heterogeneous variance-covariance structures for repeated measures. *Journal of Agricultural, Biological, and Environmental Statistics*. 1:362–389.

Yates, F. 1935. Complex experiments. *Journal of the Royal Statistical Society, Series B*. 2:181–247.