NIASRA

NATIONAL INSTITUTE FOR APPLIED STATISTICS RESEARCH AUSTRALIA

National Institute for Applied Statistics Research Australia

University of Wollongong, Australia

Working Paper

22-13

The Analysis of Tree Breeding Data using ASReml-R

Brian Cullis, David Butler, Sue Welham, Alison Smith,
Beverley Gogel and Robin Thompson

National Institute for Applied Statistics Research Australia, University of Wollongong,
Wollongong NSW 2522, Australia Phone +61 2 4221 5076, Fax +61 2 4221 4998.
Email: karink@uow.edu.au

The analysis of tree breeding data using ASReml-R

Brian Cullis
University of Wollongong
email:bcullis@uow.edu.au

David Butler
Dept. of Agriculture, Fisheries and Forestry
email: david.butler@daff.qld.gov.au

Sue Welham
VSN International
email: sue.welham@vsni.co.uk

Alison Smith
University of Wollongong
email:alismith@uow.edu.au

Beverley Gogel
University of Adelaide
email: beverley.gogel@adelaide.edu.au

Robin Thompson
Rothamsted Research
email: robin.thompson@rothamsted.ac.uk

July 15, 2013

Contents

Preliminaries

In this two-day course we give a brief overview of linear mixed models and their specification and fitting by the **ASReml-R** function implemented in R.

We only provide a sketch of the theory underpinning estimation and inference in linear mixed models, except where directly relevant, as this can be found elsewhere. Instead, after introducing the package basics, we focus on analysis of real examples kindly provided by Radiata Pine Breeding Company (RPBC).

The ASReml family of software, ie **ASReml-R** and **ASReml-SA** share the same core, however each platform offers users a choice of front-end to the core, depending mostly on their preferred working environment. We find that the R statistical computing environment to be a powerful and highly extensible environment, but its use does come at a cost for many users who are not comfortable in a command driven environment. Tools are becoming available for users who prefer a GUI environment, such as RStudio, although we do not use this package in this course, preferring the less friendly but very extensible Emacs environment. Those who wish to use the Rstudio environment should feel free to do so and copies are distributed on the course software, notes and scripts thumb drives.

Detailed documentation of the `asreml` class and package can be found in the manual (Butler et al., 2009a). This is file asreml-R.pdf located in directory *doc* in the *asreml* library directory. Note that the R command `.libPaths()` can be used to identify possible locations for this library. The manual can be accessed from within R using the command `asreml.man()` once the *asreml* library has been loaded via *library(asreml)*. Information on the version of the *asreml* library present can be obtained using the command `asreml.About()`.

The **ASReml-R** package requires a license to call the ASReml core functions. A trial license is available for workshop participants and has been distributed by email. Details of the license installed are included as a component of the `asreml` object (see Section 1.2.9).

1 The linear mixed model

In this chapter we introduce the class of linear mixed models covered in this course and the **ASReml-R** package, asreml, we will use to fit them. We start by introducing the generic form of the linear mixed model (Section 1.1) and consider 2 different parameterizations of this model. The function `asreml()` switches between these parameterizations, depending on the context, which we explain (Section 1.1.8). We then present the asreml package and discuss how different aspects of the model are specified. We give a brief overview of the `asreml` class of functions and its associated methods (Section 1.2.9), provide some supplementary information regarding types and combinations of variance models within the random model or residual error term (Section 1.2.5) and we finish Chapter 1 with an example to illustrate some of the basic features of the syntax and output.

1.1 The linear mixed model

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

$$\boldsymbol{y} = \boldsymbol{X}\boldsymbol{\tau} + \boldsymbol{Z}\boldsymbol{u} + \boldsymbol{e} \tag{1.1.1}$$

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

1.1.1 Sigma parameterization of the linear mixed model

Model (1.1.1) is called a linear mixed model or linear mixed effects model. It is assumed

$$\begin{bmatrix} \boldsymbol{u} \\ \boldsymbol{e} \end{bmatrix} \sim N\left(\begin{bmatrix} \boldsymbol{0} \\ \boldsymbol{0} \end{bmatrix}, \begin{bmatrix} \boldsymbol{G}(\boldsymbol{\sigma}_g) & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{R}_v(\boldsymbol{\sigma}_r) \end{bmatrix} \right) \tag{1.1.2}$$

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 $\boldsymbol{\sigma}_r$, and we assume that the random effects $\boldsymbol{u}$ and residual errors $\boldsymbol{e}$ are uncorrelated. The variance matrix for $\boldsymbol{y}$ is then of the form

$$\text{var}\,(\boldsymbol{y}) \quad = \quad \boldsymbol{Z}\boldsymbol{G}(\boldsymbol{\sigma}_g)\boldsymbol{Z}^{\mathsf{T}} + \boldsymbol{R}_v(\boldsymbol{\sigma}_r) \tag{1.1.3}$$

which we will refer to as the *sigma parameterization*, and the individual variance 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.

These concepts are now illustrated using the following simple (in fact the simplest) linear mixed model.

Example 1.1 A simple example

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

$$\text{var}\,(\boldsymbol{y}) \quad = \quad \sigma_u^2 \boldsymbol{Z}\boldsymbol{Z}^\intercal + \sigma_e^2 \boldsymbol{I}_n \tag{1.1.4}$$

This model has two variance 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 (1.1.3), 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* $\boldsymbol{R}_v(\boldsymbol{\sigma}_r) = \sigma_e^2 \boldsymbol{I}_n$*.*

□

1.1.2 Partitioning the fixed and random model terms

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

1.1.3 G structure for the random model terms

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

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

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

The default assumption is that each random model term generates one component of this direct sum (then $b' = b$ and $\text{var}\,(\boldsymbol{u}_i) = \boldsymbol{G}_i$ for $i = 1 \ldots b$). This means that the random effects from any two distinct model terms are uncorrelated. However, in some models, one component of $\boldsymbol{G}$ may apply across several model terms, for example, in random coefficient regression where the random intercepts and slopes for subjects are correlated, see 1.2.4. To accommodate these cases, one component of $\boldsymbol{G}$ may apply across several model terms (then $b' < b$).

Example 1.2 Variance components mixed models

Building example 1.1 to a linear mixed model with more than one (specifically $b > 1$) random effect (typically known as a variance components mixed model), the random effects $\boldsymbol{u}_i$ in $\boldsymbol{u}$, and the residual errors $\boldsymbol{e}$, are assumed pairwise uncorrelated and to each be normally distributed with mean zero and variance given by

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

and

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

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

$$\text{var}(\boldsymbol{y}) = \sum_{i=1}^{b}\sigma^2_{u_i}\boldsymbol{Z}_i\boldsymbol{Z}_i^\intercal + \sigma^2_e\boldsymbol{I}_n. \tag{1.1.5}$$

□

1.1.4 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 $\boldsymbol{e} = [\boldsymbol{e}_1^\intercal, \boldsymbol{e}_2^\intercal, \ldots, \boldsymbol{e}_s^\intercal]^\intercal$ assuming that the data vector is ordered by section, and where $\boldsymbol{e}_j$ represents the vector of errors for the j^{th} section.

1.1.5 R structure for the residual error term

For the above partitioning of $\boldsymbol{e}$ we assume that $\boldsymbol{R}_v$ is given by

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

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

For many examples presented in this course, there will be only one section.

Typically variance structures need to be specified for each random model term and often more complex models than the simple variance components model are required or desired. **ASReml-R** offers a wide range of variance models to choose from.

Separability

The concept of separability has been used extensively in multivariate analysis of variance and was described by Martin (1979) in the context of lattice processes. Martin (1979) showed that the correlation matrix of a linear-by-linear process observed on a $r \times c$ rectangular lattice can be written as the kronecker (or direct) product of two correlation matrices which relate to the rows and columns of the lattice. Both formulations are used frequently in the context of linear mixed models. To illustrate our approach, we consider the variance model for a compound model term (following the definitions of McCullagh & Nelder (1994)) which is represented by `A:B`, where the component factors `A` and `B` have n_A and n_B levels respectively and the ":" operator forms a factor which has levels equal to the combinations of all levels of `A` with all levels of `B`, and with the levels of `B` nested in the levels of `A`. Explicitly, if `A` has 3 levels and `B` has 2 levels, then `A:B` would have 6 levels, given by $1 = a_1b_1, 2 = a_1b_2, 3 = a_2b_1$ and so on. Imposition of the separability assumption leads to the following form for the variance matrix of $\text{vec}(\boldsymbol{U}_{A:B}) = \boldsymbol{u}_{A:B} = (u_{1,1}\, u_{1,2} \ldots u_{1,n_B}\, u_{n_A,n_B-1}\, u_{n_An_B})^{\mathsf{T}}$:

$$\text{var}\,(\boldsymbol{u}_{A:B}) = \boldsymbol{V}_A \otimes \boldsymbol{V}_B$$

so that the covariance between individual random effects is

$$\text{cov}\,(u_{ij},\ u_{kl}) = [V_A]_{i,k} \times [V_B]_{j,l}$$

ie. the covariance is equal to that generated by model $\boldsymbol{V}_A$, between levels i and k of factor `A` (ignoring factor `B`), multiplied by the covariance generated by model $\boldsymbol{V}_B$, between levels j and l of factor `B` (ignoring factor `A`).

The assumption of separability underpins variance model specification and construction in **ASReml-R**. In general, each of the component matrices of the R and G structures may be kronecker products of several sub-components, with each sub-component relating to simple model terms which are the components of a compound model term. These ideas will be illustrated in detail in the following chapters.

The assumption of separability also greatly reduces the computational load and contributes to the efficiency of the **ASReml-R** core, (Gilmour et al., 2009). Furthermore, separability allows a flexible framework for modelling variance structures in the linear mixed model that is genuinely appropriate in many situations. Two examples in which separable variance structures are common are as follows:

Multivariate analysis In multivariate analysis of several traits, it is common to assume independence between subjects, but to allow correlation across traits at all levels of the structure. At the residual level, this can be generated by a separable structure defined using the consolidated model term `id(subject):us(trait)` (see Section 1.2.6). Analogous definitions can be used at

other levels of the structure. In addition, correlation between subjects, eg. as a result of genetic relationships, can be applied using a suitable variance model function for the `subject` term.

Spatial analysis of a field trial Field trials are often laid out on a grid pattern (rows × columns), with various management operations being aligned with either rows or columns of the grid. It is natural to assume this may induce correlations across rows within columns and vice versa, leading to a separable correlation structure defined by `ar1(row):ar1(col)`. Note that this correlation structure can be converted into a variance structure (common variances) by changing one of the variance models `ar1` into `ar1v`; this can arbitrarily be applied to either component of this term (see Section 1.2.6).

1.1.6 Gamma parameterization for the linear mixed model

The sigma parameterization of model (1.1.3) is one possible parameterization of $\text{var}(\boldsymbol{y})$. In this parameterization both $\boldsymbol{G}(\boldsymbol{\sigma}_g)$ and $\boldsymbol{R}_v(\boldsymbol{\sigma}_r)$ are variance matrices and the variance parameters in $\boldsymbol{\sigma}_g$ and $\boldsymbol{\sigma}_r$ are referred to as *sigmas,* see above. Other parameterizations are possible and are sometimes useful. For example, in some of the early development of REML for the variance components mixed models, the variance matrix was parameterized as the equivalent model

$$\text{var}(\boldsymbol{y}) \quad = \quad \sigma_e^2 \left(\sum_{i=1}^{b} \gamma_{g_i} \boldsymbol{Z}_i \boldsymbol{Z}_i^{\mathsf{T}} + \boldsymbol{I}_n \right) \tag{1.1.6}$$

for γ_{g_i} being the ratio of the variance component for the random term $\boldsymbol{u}_i$ relative to error variance, that is, $\gamma_{g_i} = \sigma^2_{u_i}/\sigma^2_e$. Using this parameterization, the parameter σ^2_e can be "profiled" out of the iterative estimation process and this can improve the stability and speed of convergence of the average information algorithm as implemented in `asreml()`. As well, it is often simpler to provide starting values for the ratios γ_{g_i} rather than in terms of the variance components $\sigma^2_{u_i}$. This parameterization has therefore become popular and is the current default parameterization for fitting univariate, single section linear mixed models using `asreml()`. 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 (1.1.2)

$$\begin{bmatrix} \boldsymbol{u} \\ \boldsymbol{e} \end{bmatrix} \sim N \left(\begin{bmatrix} \boldsymbol{0} \\ \boldsymbol{0} \end{bmatrix}, \sigma^2_e \begin{bmatrix} \boldsymbol{G}(\boldsymbol{\gamma}_g) & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{R}_c(\boldsymbol{\gamma}_r) \end{bmatrix} \right) \tag{1.1.7}$$

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

$$\text{var}(\boldsymbol{y}) \quad = \quad \sigma^2_e \left(\boldsymbol{Z}\boldsymbol{G}(\boldsymbol{\gamma}_g)\boldsymbol{Z}^{\mathsf{T}} + \boldsymbol{R}_c(\boldsymbol{\gamma}_r) \right). \tag{1.1.8}$$

We refer to this as the *gamma parameterization*, and to the individual variance parameters in $\boldsymbol{\gamma}_g$ and $\boldsymbol{\gamma}_r$ as *gammas*. The current release of **ASReml-R** allows users to switch between the sigma and gamma parameterizations for estimation. We note here, however, that the sigma parameterization remains the more natural scale for variance components and that reporting estimates for both the sigma and gamma often confuses users. We are therefore rethinking our default approach for specifying and fitting variance models for univariate, single section data-sets and this will be implemented with the next release of **ASReml-R**. This is discussed further in Section 1.1.8.

1.1.7 Parameter types

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

1.1.8 Which parameterization does ASReml use: sigma or gamma?

The current release of **ASReml-R** default switches between the sigma and the gamma parameterizations depending on the type of data. The current default for univariate, single section data-sets is the gamma parameterization. It is possible to over-ride this default and this will be discussed further in the oats example in this chapter (see Section 1.4) and the examples in the following chapters.

The `asreml` function provides tools for viewing and reporting estimates for the both sigmas and gammas. The `summary.asreml` method is one such tool and its use will be described and illustrated in detail. Unfortunately it reports two columns, with misleading column headings when the sigma scale is invoked. This will be addressed in future versions.

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

1.1.9 Estimation and computing algorithm in `asreml`

Fitting the linear mixed model given by equation 1.1.1 involves two closely linked processes, that is, *i.* estimation of the variance parameters $(\boldsymbol{\sigma}_g^\intercal, \boldsymbol{\sigma}_r^\intercal)$ and *ii.* estimation of the fixed effects $(\boldsymbol{\tau})$ and prediction of the random effects $(\boldsymbol{u})$ for given values of the variance parameters. **ASReml-R** combines these in an implementation of the average information (AI) algorithm to obtain residual maximum likelihood (REML) (Patterson & Thompson, 1971) estimates of the variance parameters and empirical best linear unbiased estimates (BLUEs) and best linear unbiased predictors (BLUPs) of the fixed and random effects, respectively. The AI algorithm is a computationally efficient iterative algorithm to solve the REML score equations (Gilmour et al., 1995) and **ASReml-R** uses sparse matrix methods to compute the REML score and AI matrix. The computational efficiency of **ASReml-R** comes from the use of sparse matrix methods along with equation ordering to solve the mixed model equations. These equations are repeatedly computed and solved during the estimation process.

The mixed model equations (Robinson, 1991) are given by

$$\begin{bmatrix} \boldsymbol{X}^\intercal \boldsymbol{R}^{-1} \boldsymbol{X} & \boldsymbol{X}^\intercal \boldsymbol{R}^{-1} \boldsymbol{Z} \\ \boldsymbol{Z}^\intercal \boldsymbol{R}^{-1} \boldsymbol{X} & \boldsymbol{Z}^\intercal \boldsymbol{R}^{-1} \boldsymbol{Z} + \boldsymbol{G}^{-1} \end{bmatrix} \begin{bmatrix} \hat{\boldsymbol{\tau}} \\ \tilde{\boldsymbol{u}} \end{bmatrix} = \begin{bmatrix} \boldsymbol{X}^\intercal \boldsymbol{R}^{-1} \boldsymbol{y} \\ \boldsymbol{Z}^\intercal \boldsymbol{R}^{-1} \boldsymbol{y} \end{bmatrix}, \tag{1.1.9}$$

which can be written succinctly as

$$\boldsymbol{C}\tilde{\boldsymbol{\beta}} = \boldsymbol{W}^{\intercal}\boldsymbol{R}^{-1}\boldsymbol{y} \tag{1.1.10}$$

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

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

Since, in general, $\boldsymbol{X}$ is not full rank, then $\boldsymbol{C}$ is singular. A priori imposition of constraints cannot be guaranteed to cope with singularities. A call to `asreml` provides a non-unique solution to equation 1.1.10 using a generalised inverse of $\boldsymbol{C}$, and this solution is given by

$$\tilde{\boldsymbol{\beta}}_0 = \boldsymbol{C}^{-}\boldsymbol{W}^{\intercal}\boldsymbol{R}^{-1}\boldsymbol{y}$$

where $\boldsymbol{C}^{-}$ is a reflexive generalised inverse of $\boldsymbol{C}$ (ie. $\boldsymbol{C}^{-}\boldsymbol{C}\boldsymbol{C}^{-} = \boldsymbol{C}^{-}$ and $\boldsymbol{C}\boldsymbol{C}^{-}\boldsymbol{C} = \boldsymbol{C}$).

1.2 Specifying and fitting linear mixed models in `asreml`

1.2.1 An overview of the model formula syntax

Linear mixed models can be succinctly represented in text (and in software such as R) by extending the symbolic model formulae of Wilkinson & Rogers (1973). The full linear mixed model in `asreml` is described with 4 formula objects:

`fixed` to specify the fixed effects terms,

`random` to specify the random effects,

`rcov` to specify the residual covariance structure, and

`sparse` to specify the fixed effects terms for which standard errors and Wald tests are not required. The equations for the effects from these terms are included using sparse matrix methods and inclusion of ancillary terms in this argument can therefore decrease execution time. The factor `mv`, which is a reserved model term (see Table 1.1) is included automatically in this model formula.

These model formulae contain terms which are constructed using the Wilkinson and Rogers syntax in which model terms are separated by a "+" operator, and model terms are constructed from individual variables, which in general may be factors or variates, Interactions between factors or variates are formed with the ":" operator in the R convention. Additional operators, such as "*" and "/" (crossing and nesting operators as defined by Wilkinson and Rogers) can also be used. For example, for factors `A` and `B`, these operators are defined as `A*B = A + B + A:B` and `A/B = A + A:B`, where `A:B` is a model term which consists of all combinations of levels from the factors `A` and `B`. The interpretation of `A:B` depends on which other terms are fitted in the model. For example, it is the interaction between `A` and `B` if both main effects of `A` and `B` have been fitted in the model but it is the effects of `B` within `A` if only the main effect of `A` is fitted in the model. In other words,

```
fixed = y ~ A*B
```

is the syntax for fitting a model where `A:B` defines the interaction between `A` and `B`, while

```
fixed = y ~ A/B
```

is the syntax for fitting a model where `A:B` defines the effects of `B` within `A`.

Identifiers for variates and factors can be any legal R identifier, except for several reserved `asreml` names. These are presented in Table 1.1. (see section 1.2.9 for a detailed description of arguments such as `y`, `data`, `na.method.Y` and so on which are mentioned in this table).

Table 1.1: List of reserved names used in `asreml`.

Reserved name	Model Fixed/random	Description
`mv`	Fixed	includes a factor with number of levels equal to the number of missing values of the response specified in the `y` argument of `asreml`. Missing values in `y` are handled according to the argument `na.method.Y`. If `na.method.Y='omit'` records with missing values are deleted; if `na.method.Y='included'` records with missing values are included and a level is generated for each missing value in the internal factor `mv`.
`trait`	Either	Used with multivariate `Y` to fit an intercept for each column in `Y`. It can also be used to form interactions with other model terms as is usually required in multivariate analyses. It is the multivariate analogue of `1`.
`units`	Random	A factor with a level for each record; allows additional error terms to be included.

1.2.2 Constructor functions for model terms

Fixed model terms are used to construct the design matrix $\boldsymbol{X}$ and associated effects $\boldsymbol{\tau}$, and random model terms are used to construct the design matrix $\boldsymbol{Z}$ and associated effects $\boldsymbol{u}$, as given in equation (1.1.1).

The `asreml` model syntax allows model terms to be created by applying functions to the variables comprising the fixed or random model terms to modify the form of the term. We call these *constructor functions*, as they are used to modify the construction of the design matrices $\boldsymbol{X}$ and $\boldsymbol{Z}$. The most common constructor functions are listed in Table 1.2; full details of these and additional functions can be found in the **ASReml-R** reference manual (Butler et al., 2009b).

Table 1.2: List of common constructor functions used in `asreml` (`x` is a variate, `A` is a factor).

Function name	Model Fixed/random	Description
`dev(x)`	Either	Factor version of variate `x` with a level for each `unique(x)`.
`lin(A)`	Either	Variate version of factor `A` (using ordinal levels)
`pol(x,t)`	Either	Orthogonal polynomials of order $(0 : \|t\|)$ generated from `x`, if $t < 0$ then order 0 polynomial (ie. intercept) is omitted.
`spl(x, k, points)`	Random	Non-linear component of a cubic spline for variate `x`. Usually used in conjunction with `x` or `lin(x)` fitted as fixed effects. Knots points for the cubic spline are placed at the design points if `length(unique(x))` $< k$ otherwise there are k equally spaced knot points over the `range(x)`. The default for k is 50. Alternately, `points` may contain a vector of user specified knot points. Both may be omitted and defaults set in `asreml.control`.
`at(A,l)`	Either	Condition on `l`th level of factor `A`, where `l` may be a scalar or a vector of levels, either as numeric or character. Note that if `l` is numeric, then the level of `A` is chosen as the `l`th in the factor (sorted) order. Note also that when used with spline terms, such as `at(A,2):spl(x)` then the knot points are derived from all of the factor `A`, not just level 2.
`grp(obj)`	Either	Groups contiguous columns in the user supplied data-frame provided to the `data` argument, to be treated as a single model term named "`obj`". The columns of `data` are identified by a character or numeric vector component `obj` of the `group` argument to `asreml.control`.
`mbf(obj)`	Either	Includes `obj` as a set of columns to be fitted as a single model term similar to `grp`. The name `obj` must also appear as a component of the `mbf` argument to `asreml.control` where the data-frame holding the columns is identified along with a key field for merging records with those in the data-frame specified in the `data` argument to `asreml`.
`and()`	Either	Allows construction of non-standard design matrices

1.2.3 Applying variance models to random model terms

The random model terms define the design matrix $\boldsymbol{Z}$ and associated set of random effects, but additional information is required for this term before model fitting can be achieved. This additional step involves specifying the G structure for each term. In general, there are two different approaches to this problem, which we will call the functional and structural approaches. The structural approach applies variance models to individual terms after the random model has been defined; this is the approach taken in the **ASReml-SA** (Gilmour et al., 2009)) prior to release 4. This gives a step-by-step approach to building up the full model that is straightforward but not concise. In contrast, **ASReml-R** has extended the Wilkinson and Rogers syntax allow specification of the variance models directly to each model term. This is called the functional approach, as it uses special model functions to directly apply variance models to individual or pre-chosen model term(s) in the random model formula. This process forms what we refer to in the following as a consolidated model term that simultaneously defines both the design matrix ($\boldsymbol{Z}_i$) and variance model ($\boldsymbol{G}_i$). This process is shown in Table 1.3 and some common variance functions are defined in Table 1.4. The full range of variance model functions and their detailed definition can be found in the **ASReml-R** manual (Butler et al., 2009b). Note that the variance model function name is presented following the R convention of including the brackets with the function name to denote that the `name` is a function.

Table 1.3: Building a consolidated model term in **ASReml-R**.

Model no.	Model term	Simple components	Variance model function name	Covariance component	Consolidated model term
1∗	`A:B`	`A` `B`	`id()` `id()`	`id(A)` `id(B)`	`id(A):id(B)`
2	`A:B`	`A` `B`	`idv()` `id()`	`idv(A)` `id(B)`	`idv(A):id(B)`
3	`A:B`	`A` `B`	`id()` `idv()`	`id(A)` `idv(B)`	`id(A):idv(B)`
4∗	`A:B`	`A` `B`	`id()` `ar1()`	`id(A)` `ar1(B)`	`id(A):ar1(B)`
5	`units:trait`	`units` `trait`	`id()` `us()`	`id(units)` `us(trait)`	`id(units):us(trait)`
6	`A:B`	`A` `B`	`fa()` `id()`	`fa(A)` `id(B)`	`fa(A,k=1):id(B)`

Variance models can *only* be applied to simple model terms, or simple model terms with a

constructor function. The underlying principle in the construction of variance models is that of separability (see section 1.1.5) and this governs the manner in which the variance model ($\boldsymbol{G}_i$) is constructed. This approach can lead to a lack of symmetry as illustrated in models 2 and 3 of Table 1.3 where the `idv()` and `id()` variance functions could be interchanged without altering the overall form of the variance model for `A:B`. An alternate, but less concise variance model function which avoids this lack of symmetry is presented later (see Table 1.5). Models marked with an asterisk have variance models that are *correlation* models.

Table 1.4: List of common variance model functions, their type (correlation or variance) and a brief description.

Function name	Type	Description
`id()`	Correlation	IID with variance 1
`idv()`	Variance	IID with common variance
`idh()`	Variance	independent with separate variances
`ar1()`	Correlation	auto-regressive structure of order 1
`cor()`	Correlation	unstructured correlation matrix
`exp()`	Correlation	exponential power model (based on distances)
`diag()`	Variance	independent with separate variances (same as `idh()`)
`giv()`	(Known) correlation	Applies a known (scaled) correlation (or inverse correlation) matrix; the number of rows in the matrix must be `length(levels(obj))` and the order is assumed correct
`giv(,ped=T)`	(Known) correlation	Applies a known (scaled) inverse correlation matrix typically derived from a pedigree file with the factor argument; there may be more rows in the inverse matrix than levels of the argument (see **asreml.Ainverse**).
`us()`	Variance	general unstructured, symmetric positive definite covariance matrix
`fa(,k)`	Variance	factor analytic model of order k

In Table 1.4, the correlation models take value 1 on the diagonal but variance models can take any positive value on the diagonal. As shown in Table 1.3, correlation variance models can be appended with `v` (eg. `idv()`) to add a common (homogeneous) variance or with `h` (eg. `idh()`) to add a separate (heterogeneous) variance for each level of the factor. For these models, the parameters associated with the correlation model are scale-independent, and those associated with the common or heterogeneous variances are scale-dependent.

If the consolidated model term definition is incomplete, ie. if some (or all) of the variables do not have a variance model specified, the default variance model, `id()`, is applied to these variables.

1.2.4 Extended variance models

Table 1.5 presents the extended variance model functions currently available in `asreml`. The `dsum` variance function is not yet implemented but will be available in version 4. Hence, for now, we illustrate the use of `str`.

Table 1.5: List of extended variance model functions, their type (correlation or variance) and a brief description.

Function name	Argument	Description
`str`	`form`	a model formula specifying a set of model terms to be included in the random model formula, which will collectively have the associated `vmodel` variance model
	`vmodel`	a formula object containing valid variance functions separated by ":" operators specifying the G-structure which applies to the set of terms in `form`. The size of the variance structure(s) can be given as an interger argument in place of the usual *factor* object.
`dsum`	`model`	is any valid model formula and a conditioning factor separated by the "|" symbol.
	`levels`	A list of length the number of (consolidated) model terms separated by "+", the elements of which are vectors containing the levels of the conditioning factor
	`outer`	Still under development - details to come

Imposing structure across random model terms

The extended variance model function `str()` can be used to make a component of the direct-sum G structure apply across several model terms. This is most commonly required for random coefficient regression models, to enable correlation between random subject intercepts and slopes. In the simplest case, with a set of 5 subjects (factor `Subject`) with explanatory variate `x`, this is specified as

```
random = ~ str(form=~ Subject + Subject:x, vmodel=~ us(2):id(5))
```

The random model formula is constructed to ensure that the model terms specified by the argument `form` are kept together (here `Subject` followed by `Subject:x`). The variance structure defined using the `vmodel` argument begins at the start of the first term specified and is expected to exactly span the whole set of terms given in argument form. 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 to the variance model required. In this context, the size of the variance structures can be given as an integer argument to the variance functions, as a suitable factor is usually not available.

In our example, this generates a combined set of random effects from the individual subject intercepts, $\boldsymbol{u}_I = (u_{I1} \dots u_{I5})^\intercal$ and subject slopes, $\boldsymbol{u}_S = (u_{S1} \dots u_{S5})^\intercal$, as $\boldsymbol{u}_{IS} = (\boldsymbol{u}_I^\intercal \, \boldsymbol{u}_S^\intercal)^\intercal$. The consolidated term then has a variance structure of the form

$$\text{var}\,(\boldsymbol{u}_{IS}) = \text{var}\left(\begin{bmatrix}\boldsymbol{u}_I \\ \boldsymbol{u}_S\end{bmatrix}\right) = \begin{pmatrix}\sigma_{II} & \sigma_{IS} \\ \sigma_{IS} & \sigma_{SS}\end{pmatrix} \otimes \boldsymbol{I}_5 = \begin{pmatrix}\sigma_{II}\boldsymbol{I}_5 & \sigma_{IS}\boldsymbol{I}_5 \\ \sigma_{IS}\boldsymbol{I}_5 & \sigma_{SS}\boldsymbol{I}_5\end{pmatrix}$$

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 $\sigma_{IS}/\sqrt{\sigma_{II}\sigma_{SS}}$).

1.2.5 Identifiability

Once all variables have a variance model function applied, `asreml` attempts to determine whether the term is identifiable, that is, if it can be separately estimated from (is not confounded with) other terms in the model. If the consolidated term generates a correlation matrix, for example, the consolidated model term for `A:B` is specified as `id(A):ar1(B)`, then it is usually the case that one wishes to fit a model with this specified correlation structure but to also allow the effects have a common variance. As already noted, when a correlation structure is specified for the residual error term, `asreml` will detect this and add a common variance. In general, it is necessary for the user to complete the specification. For example, `id(A):ar1(B)` should become either `idv(A):ar1(B)` or `id(A):ar1v(B)`; it is arbitrary which variable the common variance is attached to. If more than one variance model function in the consolidated model term generates a variance structure (either homogeneous or heterogeneous), for example `idv(A):ar1v(B)`, then the parameters will not all be identifiable and so the user must either change `idv(A)` to `id(A)` and leave `ar1v(B)` as it is, or change `ar1v(B)` to `ar1(B)` and leave `idv(A)` as it is.

A consolidated model term thus consists of a set of variables, each with a variance model function applied. This generates a *separable* variance structure for the term.

1.2.6 Applying variance structures to the residual error term

The residual error term is also defined using a consolidated model term which is specified as a formula in the `rcov` argument to `asreml`. For example, for the default situation of *IID residual errors*, the `rcov` statement would be

```
rcov=~idv(units)
```

Alternatively, the following code would specify a *separable autoregressive spatial model of order 1 (AR1×AR1)* for the observations from a trial arranged in a rectangular array indexed by the data variables `column` and `row`

```
rcov=~ar1v(column):ar1(row)
```

To apply this variance structure the observations would need to cover the whole grid, and it would be necessary to pre-order the data file as rows within columns.

1.2.7 Special properties and rules in defining the residual error term

There are certain properties and rules in specifying the residual error term that require special consideration:

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

Rule 2 Where a compound model term is specified for the residuals, each combination of levels of the simple model terms comprising this term must uniquely identify one unit of the data.

Rule 3 The data *must* be ordered to match the R structure specified.

These rules will always be satisfied for a single section of data, that is, $\boldsymbol{R}_v = \boldsymbol{R}_{v_1} = \sigma^2 \boldsymbol{I}_n$, see Example 1.2. 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.

We will now consider these issues in the context of field trial analysis. We first consider the analysis of a single trial and move from there to the analysis of multiple trials.

Analysis of a single field trial Consider a field trial comprising 4 replicates of 24 varieties with a grid layout of 4 rows and 24 columns (rows are replicates). The data frame takes the form:

```
Trial  Row  Column Variety yield
1      1    1      C        5.43
1      1    2      M        6.01
1      1    3      J        6.31
.
.
.
1      4    23     R        4.22
1      4    24     B        4.89
```

where *Trial*, *Row*, *Column* and *Variety* are factors and *yield* is a variate. To fit a separable autoregressive spatial model (order 1), we would specify the residual model formula

```
rcov=~ar1(Row):ar1(Column)
```

The ordering of the two terms is important: this specification means that the errors are ordered as columns within rows. As there is no design matrix to mediate between the data and the vector of residual errors, it is necessary for the data to be ordered as columns within rows, as per the data frame above (Rule 3). In the `asreml` function, this is checked and a message is printed if the ordering of the data appears incompatible with the specified model.

An error will also be given if the number of data units is not equal to the number of effects generated by the direct product structure, ie. the product of the number of levels in the factors used to construct the term (Rule 1 and Rule 2). If this is the case, the term specified cannot be used as the residual error term.

Note that the `rcov` formula is defined as a correlation model and, as stated earlier, `asreml` will automatically add a common variance in this situation and estimation is achieved using the gamma parameterization.

Analysis of data with multiple sections Now consider the analysis of a series of 12 field trials testing the same 24 varieties, such that all trials have a grid layout but differ in their dimension: trials 1-8 are 4×24 and trials 9-12 are 8×12. The data for all 12 trials is contained in a single data frame and is ordered first by `Trial`, and then by columns within rows within each trial. To fit a separable auto-regressive model (order 1) within each trial, we would specify the residual model formula

```
rcov=~at(Trial):ar1v(Row):ar1(Column)
```

To fit a separable auto-regressive model (order 1) within each trial but with spatial correlation in the column direction only for trials 1-8 (and independence across rows for these trials), we would specify the residual model formula

```
rcov=~at(Trial,c(1:8)):idv(Row):ar1(Column)+at(Trial,c(9:12)):ar1v(Row):ar1(Column)
```

1.2.8 Using `at()` in specifying the residual model term for data with sections

In the current version of **ASReml-R**, the previous example highlights the duality and potential confusion in the use of `at()` for specification of the residual variance model for the analysis of multiple trials. The `at` function was described in table 1.2 as a constructor function that provides a way of constructing a model term which allows conditioning on a factor, and it can be used in either the *fixed* or *random* model formulae. In the context of an `rcov` definition, the `at()` function performs several different tasks. The underlying `Trial:Row:Column` term informs `asreml` that we are working with a direct product of the `Trial`, `Row` and `Column` factors. The use of `at(Trial)` modifies this to a direct sum structure, with the different components of the direct sum (sections) applying to units with different levels of the factor `Trial`. The use of `at(Trial)` in this context also means that the `Row:Column` compound model terms are pruned within trials (so that only the levels used within each trial apply to that trial) and that a separate residual variance is applied to each trial. Again, where the within-section models are defined as correlation matrices (see single field trial example), `asreml` will automatically add a common variance within each section.

Analysis of data with multiple sections continued... For each section (here trials), it is necessary that the size of the direct product structure generated (after pruning) matches the number of units in the section (Rule 2), and that the ordering of the units within each section matches that expected for the direct product structure (here, ordered as columns within rows for each

trial, Rule 3).

This context-dependent behaviour of the `at()` *function is undesirable as it is potentially confusing and this will be resolved in the next release of* asreml, *where it will revert to being used only as a constructor function. This is one of the motivations behind the development of the new extended function* `dsum()` *will is aimed to resolve this duality and provide a more seamless and extensible approach to forming consolidated model terms which are direct sums of more than one model term, possibly conditioned on a factor or set of factors. This new extended model function will be available for use in either* `random` *or* `rcov`

For multiple sections we recall that ***ASReml-R*** *uses the sigma parameterization, although the* `summary.asreml` *function will produce two columns with misleading headings.*

1.2.9 The `asreml` class in more detail

In addition to the model formulae introduced above, there are several other arguments to `asreml` that provide additional information and user-control on the fitting of the linear mixed model. Table 1.6 outlines some of the arguments to `asreml` and the corresponding components of the linear model or methodology to which they refer (where applicable).

The function `asreml` returns an object (of class `'asreml'`) which is a list that contains the results for the linear mixed model specified in the call to `asreml()`. Objects with the `'asreml'` class have methods given by `plot`, `summary`, `predict`, `fitted`, `coef`, `update`, `residuals`, `tr`, `variogram`, `wald` and `plot.asrVariogram`. Related functions include `asreml.Ainverse`, `asreml.control`, `asreml.constraints`, `asreml.asUnivariate`, `asreml.man`, `asreml.read.table` and `asreml.variogram`. Some of these methods and functions will be briefly described and illustrated in the following chapters.

The most frequently used components of an object of class `asreml` are:

`loglik` The REML log-likelihood on exit.

`gammas` The (current) estimates of $\boldsymbol{\gamma}_r$ and $\boldsymbol{\gamma}_g$ if the gamma parameterization is used, otherwise.

`coefficients` A list with three components, `fixed, random` and `sparse` containing non-unique E-solutions for `fixed` and `sparse` model terms, and E-BLUPS for `random` model terms. Each component is provided with meaningful `names`, where "E" stands for Empirical.

`fitted.values` A vector containing the fitted values (including random effects) from the model fit.

`residuals` A vector containing the residuals from the model fit.

`sigma2` Current estimate of σ_e^2 for gamma parameterization, else set to 1.

`ai` The inverse average information matrix of the vector of variance parameters on exit. A vector of length $nk(nk+1)/2$ where nk=`length(gammas)`, containing the lower triangle row-wise.

Table 1.6: Key arguments to `asreml()`

Argument	Value	Context
`fixed`	formula object specifying *fixed model terms*. If the response evaluates to a matrix then the reserved name `trait` with levels `dimnames(y)[[2]]` is added to the model frame	$\boldsymbol{X}$
`random`	formula object specifying *random model terms*, but without the response preceding the	$\boldsymbol{Z}$
`sparse`	formula object specifying *fixed model terms*, to be included using sparse matrix methods	$\boldsymbol{X}$
`rcov`	formula object specifying the R structure. The default is the reserved name `units` which is a factor defined as `factor(seq(1,nrow(data)))` and is automatically included in the model frame	$\boldsymbol{R}$
`data`	a *data frame* containing the variables names in `fixed, random, sparse` and `rcov`	
`family`	a family object - a list of functions and expressions for defining the link and the variance functions. The currently supported families are `gaussian, binomial, poisson, negative binomial` and `Gamma`. Use of the additional argument `dispersion` allows control of the estimation of the residual variance with default `NA` for `asreml.gaussian` making `asreml` to use the sigma parameterization	
`na.method.Y`	a character string (`'include', 'omit'` or `'fail'`) specifying how missing values in the response will be handled. Default is `'include'`.	
`na.method.X`	a character string (`'include', 'omit'` or `'fail'`) specifying how missing values in the fixed and random design matrices will be handled. Default is `'omit'`.	
`weights`	a character string or name identifying the variable in `data` to use as weights in the analysis	

`score` The REML score vector (of length nk).

`call` The function call.

1.3 More on variance models

In this supplementary section, we provide some more detailed background on different types of variance models. There are three generic types of variance model allowed in `asreml` that can

be used for R and G structures , namely, *correlation models*, *homogeneous variance models* and *heterogeneous variance models*. In the following sections, we give a formal definition of each type of model, and then discuss how they can be combined to give valid variance models with an identifiable scale.

1.3.1 Correlation models:

In correlation models all diagonal elements are identically equal to 1. If the $q \times q$ matrix $\boldsymbol{C} = [c_{ij}]$, $i, j = 1 \ldots q$, denotes the correlation matrix for a particular correlation model, then

$$\boldsymbol{C} = [c_{ij}] \quad : \quad \begin{cases} c_{ii} = 1, & \forall i \\ c_{ij} = c_{ji} & |c_{ij}| < 1,\ i \neq j. \end{cases}$$

The simplest correlation model is the identity model (`id()`) for which the off-diagonal elements are identically equal to zero, that is, $c_{ij} = 0$ for $i \neq j$.

Correlation models often arise in longitudinal data analysis (eg. `ar1()`), and in geostatistics or spatial statistics (eg. `exp()`, `mtrn()`). There are also more general correlation models such as the banded model which is a simplification of the completely general correlation model with $q(q-1)/2$ parameters.

1.3.2 Homogeneous variance models

In homogeneous variance models the diagonal elements all have the same positive value, σ^2 say. If $q \times q$ matrix $\boldsymbol{V} = [v_{ij}]$, $i, j = 1 \ldots n$ is an homogeneous variance matrix, then

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

Note that if $\boldsymbol{V}$ is the homogeneous variance model matrix corresponding to the correlation model matrix $\boldsymbol{C}$ then

$$\boldsymbol{V} \quad = \quad \sigma^2 \boldsymbol{C}$$

with $v_{ij} = \sigma^2 c_{ij}$. This model has just one more parameter than $\boldsymbol{C}$. For example, the homogeneous variance model corresponding to the identity correlation structure is the simple variance components model (`idv()`), for which $v_{ii} = \sigma^2$ for $i = 1 \ldots q$, with $v_{ij} = 0$ for $i \neq j$, ie. off diagonal elements equal to zero.

1.3.3 Heterogeneous variance models:

The third variance model is the *heterogeneous* variance model for which the diagonal elements are positive but differ. If $q \times q$ matrix $\boldsymbol{V} = [v_{ij}]$ for $i, j = 1 \ldots q$, is a heterogeneous variance matrix, then

$$\boldsymbol{V} = [v_{ij}] \quad : \quad \begin{cases} v_{ii} = \sigma_i^2, & \forall i \\ v_{ij} = v_{ji}, & i \neq j. \end{cases}$$

If $\boldsymbol{V}$ is the heterogeneous variance model matrix corresponding to the correlation model matrix $\boldsymbol{C}$, then

$$\boldsymbol{V} \quad = \quad \boldsymbol{DCD}$$

where $\boldsymbol{D}$ is a diagonal matrix with q rows, ie. $\boldsymbol{D} = \text{diag}\,(d_i)$ and $v_{ij} = d_i d_j c_{ij}$. This model has an additional q parameters compared to the base correlation model. For example, the heterogeneous variance model corresponding to the identity correlation model (`idh()` or `diag()`) specifies the diagonal variance model, with zero off diagonal elements.

Other examples include the factor analytic (`fa`) or ante-dependence `ante` variance models and the most general is the unstructured (`us`) with $q(q+1)/2$ parameters.

1.3.4 Combining variance models

There are some general principles which can be useful in avoiding over-parameterization of variance models and in the following we present some of these by way of example.

When either $\boldsymbol{R}$ or $\boldsymbol{G}$ is formed from the kronecker product of several sub-matrices some general rules must be obeyed to avoid over-parameterisation. In the following we consider models with two components for $\boldsymbol{G}$ and $\boldsymbol{R}$ and use $\boldsymbol{C}_i$ and $\boldsymbol{V}_i, i = 1, 2$ to denote arbitrary correlation and variance matrices, respectively.

1. If $\boldsymbol{R} = \boldsymbol{C}_1 \otimes \boldsymbol{C}_2$ then this is a valid correlation model and a scale parameter must be added and this is the default action in `asreml` for a univariate/single section data-set. In multi-section problems `asreml` also checks that each $\boldsymbol{R}_i$ is a variance matrix, else it will include a variance parameter to make each $\boldsymbol{R}_i$ a variance model.

2. If $\boldsymbol{G} = \boldsymbol{C}_1 \otimes \boldsymbol{C}_2$ then a scale parameter should be added to one of the correlation models. The default action is described in Section 1.2.3.

3. If $\boldsymbol{R} = \boldsymbol{C}_1 \otimes \boldsymbol{V}_2$ or $\boldsymbol{R} = \boldsymbol{V}_1 \otimes \boldsymbol{C}_2$ this defines a valid (identifiable) variance matrix and the sigma parameterization is used.

4. If $\boldsymbol{G} = \boldsymbol{C}_1 \otimes \boldsymbol{V}_2$ or $\boldsymbol{G} = \boldsymbol{V}_1 \otimes \boldsymbol{C}_2$ then $\boldsymbol{G}$ is a valid (identifiable) variance matrix and no further action is required.

5. If $\boldsymbol{R}$ or $\boldsymbol{G} = \boldsymbol{V}_1 \otimes \boldsymbol{V}_2 = \boldsymbol{V}$ then $\boldsymbol{R}$ or $\boldsymbol{G}$ is an over-parameterized (unidentifiable) variance matrix, and it is necessary to determine the scale by fixing one of the scale-dependent parameters in either $\boldsymbol{V}_1$ or $\boldsymbol{V}_2$.

Table 1.7: A split-plot field trial of oat varieties and nitrogen applications

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

1.4 Example: oats data-set

We now illustrate some of the ideas presented in the previous sections using a simple, yet informative, example. Within any R we can load the **ASReml-R** package using the command

```
> library(asreml)
```

The oats example is a split-plot experiment first presented by Yates (1935). The experiment was conducted to examine the effects of nitrogen (as manure) at 4 levels (0, 0.2, 0.4 and 0.6 cwt/acre) on the yield of three varieties of oats (Golden Rain, Marvellous and Victory). 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 per block while the four levels of nitrogen were randomly assigned to the four sub-plots. The data are presented in Table 1.7 and a graphic display is presented in Figure 1.1.

A standard analysis of these data recognises the two basic elements inherent in the experiment. These are firstly, the stratification of the experimental units (which are also the observational units), that is, the blocks, whole-plots and sub-plots, and secondly, the treatment structure that is superimposed on the experimental units. The aim of the analysis is to extract the treatment effects and examine their importance while allowing for the stratification of experimental units and the restricted randomisation of the units to the treatments.

Brien & Demetrio (2009) discuss formulating mixed models for the analysis of designed experiments. This experiment is a single phase, multi-stratum experiment in which the so-called intra-tier random formula is given by `Blocks/Wplots/Subplots`, with the treatment formula given by `Variety*Nitrogen`. The `oats` data-frame is distributed with **ASReml-R** and has six fields, given by

```
> names(oats)
[1] "Blocks"   "Nitrogen" "Subplots" "Variety"  "Wplots"   "yield"
```

The first five fields are factors with `yield` being the response. There are numerous ways to specify and fit this model to the data. Three versions are presented below:

```
oats.gammas.asr <- asreml(fixed=yield ~ Variety*Nitrogen,random=~Blocks/Wplots,
                   data=oats)
oats.vars.asr <- asreml(fixed=yield ~ Variety*Nitrogen,random=~Blocks/Wplots,
                        rcov=~idv(units),data=oats,
                        family=asreml.gaussian(dispersion=1))
oats.vars.asr <- asreml(fixed=yield ~ Variety*Nitrogen,
                        random=~idv(Blocks) + id(Blocks):idv(Wplots),
                        rcov=~idv(units),data=oats,
                        family=asreml.gaussian(dispersion=1))
```

The first call represents the most succinct way to fit the model. It is not necessary to specify the `rcov` structure as the default is `id(units)`. A variance parameter σ_e^2 is automatically added to the residual variance model, and since the data-set is a single section, univariate model **ASReml-R** uses the gamma parameterization for estimation. **ASReml-R** automatically assigns the default variance function model (`idv()`) to both of the terms in the random model formula.

Estimation of the variance parameters using the sigma parameterization can be achieved by use of the `dispersion` argument of `asreml.gaussian`. This model also requires specification of the residual variance model as a variance model otherwise no residual variance parameter will be estimated. The third model call explicitly specifies the variance models for each of the terms in the random model formula.

The estimates of the variance components from the first and last fits are:

```
> summary(oats.gammas.asr)$varcomp
                             gamma component std.error  z.ratio constraint
Blocks!Blocks.var        1.2111647  214.4771 168.83404 1.270343   Positive
Blocks:Wplots!Blocks.var 0.5989373  106.0618  67.87553 1.562593   Positive
R!variance               1.0000000  177.0833  37.33244 4.743416   Positive
> summary(oats.vars.asr)$varcomp
                            gamma component std.error  z.ratio constraint
Blocks!Blocks.var        214.4771  214.4771 168.83405 1.270343   Positive
Blocks:Wplots!Wplots.var 106.0618  106.0618  67.87553 1.562593   Positive
R!variance                 1.0000    1.0000        NA       NA      Fixed
R!units.var              177.0833  177.0833  37.33244 4.743416   Positive
```

Note the two forms of the outputs for each of the two parameterizations. Both summaries have columns with the same names, the former correctly labelling the column containing the gamams, using *component* for the column containing the sigmas, while the latter has two copies of the *sigma* parameterization. Note also that the *R!variance* line is an additional line in the second summary which is unnecessary to display and is a relic of previous approaches and should be ignored.

The general approach to testing the significance of fixed effects under REML estimation for linear mixed models is to use Wald statistics. These statistics can be produced using the `Wald`

method, which has two frequently used arguments which we now briefly discuss. Firstly, use of the `ssType` argument controls the type of test that is undertaken for each of the terms in the fixed model formula (in the order returned by `asreml`. Wald tests can be either `incremental` (the default) or `conditional`. The former produces tests from the notion of an incremental sums of squares approach in the spirit of that used in classical regression analysis (see Searle, 1971, for example). The former attempts to apply the philosophy of respecting both intrinsic and structural marginality, as discussed in detail by Nelder (1994). Nelder argues that meaningful and interesting tests for terms in linear (and linear mixed) models can only be conducted for those tests which respect marginality relations.

To illustrate these ideas we consider the current example. The incremental sums of squares for this model can be written as the sequence

$$\begin{aligned} R(1) & \\ R(\mathtt{V}|1) &= R(1+\mathtt{V}) - R(1) \\ R(\mathtt{N}|1+\mathtt{V}) &= R(1+\mathtt{V}+\mathtt{N}) - R(1+\mathtt{V}) \\ R(\mathtt{V}:\mathtt{N}|1+\mathtt{V}+\mathtt{N}) &= R(1+\mathtt{V}+\mathtt{N}+\mathtt{V}:\mathtt{N}) - R(1+\mathtt{V}+\mathtt{N}) \end{aligned}$$

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, and we use shortened names for the factors for ease of presentation. Implicit in these calculations is that

1. we only compute Wald statistics for estimable functions (Searle, 1971)

2. all variance parameters are held fixed at the current estimates from the maximal model.

The summary of this information is returned in the `Wald` component of the `list` returned by `wald.asreml`.

The second frequently used argument of `wald.asreml` is `denDF`. **ASReml-R** uses the approach of Kenward & Roger (1997) to compute approximate denominator degrees of freedom of the Wald test for each term in the fixed effects model formula, and this can be invoked using the `denDF` argument, for *moderately* sized problems. The approach used to compute these quantities is controlled by the setting of `denDF`; `'default'` being the recommended setting.

Kenward & Roger (1997) pursued the concept of construction of Wald-type test statistics through an adjusted variance matrix of var $(\hat{\boldsymbol{\tau}})$. They argued that it is useful to consider an improved estimator of the variance matrix of var $(\hat{\boldsymbol{\tau}})$ which has less bias and accounts for the variability in estimation of the variance parameters. At this stage the Wald statistics currently computed by **ASReml-R** use an unadjusted variance matrix.

The following illustrates some of these concepts for this example:

```
> wald(oats.vars.asr)
Wald tests for fixed effects

Response: yield

Terms added sequentially; adjusted for those above

                 Df Sum of Sq Wald statistic Pr(Chisq)
(Intercept)       1   245.141        245.141    <2e-16 ***
Variety           2     2.971          2.971    0.2264
Nitrogen          3   113.057        113.057    <2e-16 ***
Variety:Nitrogen  6     1.817          1.817    0.9357
residual (MS)             1.000
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
> wald(oats.vars.asr,denDF='default',trace=FALSE)$Wald
Algebraic ANOVA Denominator DF calculationis not available
Empirical derivatives will be used.

                 Df denDF    F.inc           Pr
(Intercept)       1     5 245.1000 1.931825e-05
Variety           2    10   1.4850 2.723869e-01
Nitrogen          3    45  37.6900 2.457710e-12
Variety:Nitrogen  6    45   0.3028 9.321988e-01
> wald(oats.vars.asr,denDF='default',ssType='conditional',trace=FALSE)$Wald
Algebraic ANOVA Denominator DF calculationis not available
Empirical derivatives will be used.

                 Df denDF    F.inc   F.con Margin           Pr
(Intercept)       1   8.9 245.1000 77.1700        1.040598e-05
Variety           2  10.0   1.4850  1.4850      a 2.723869e-01
Nitrogen          3  45.0  37.6900 37.6900      A 2.457710e-12
Variety:Nitrogen  6  45.0   0.3028  0.3028      B 9.321988e-01
```

The first call to `wald` provides "standard", incremental Wald tests for each of the four terms in the fixed model formula. The default reference distribution is χ^2 with degrees of freedom for each term given by the number of non-singular equations involved in the term.

The next call uses the `denDF` argument to compute the approximate denominator degrees of freedom and thence an approximate F-type Wald test is computed in place of the χ^2 statistics used in the previous ANOVA summary table. Note that the F-statistics are equivalent to the χ^2 statistics divided by the numerator degrees of freedom and the approximation for the denominator degrees of freedom is exact being a consequence of the orthogonal block design.

The final call invokes the *conditional* tests, which in this case are the same as the *incremental* tests (apart from the `Intercept`) since the design is orthogonal, but this will be the case in general. Both types of tests are displayed but there is an additional column providing the marginality status of each term with respect to other terms in the table. The letter `B` denotes

that the interaction term has been adjusted for all other terms in the table, while the lower and upper case `A` beside the main effects denotes that these terms have been adjusted for other terms with the same letter as well as those without a blank in the `Margin` column as required.

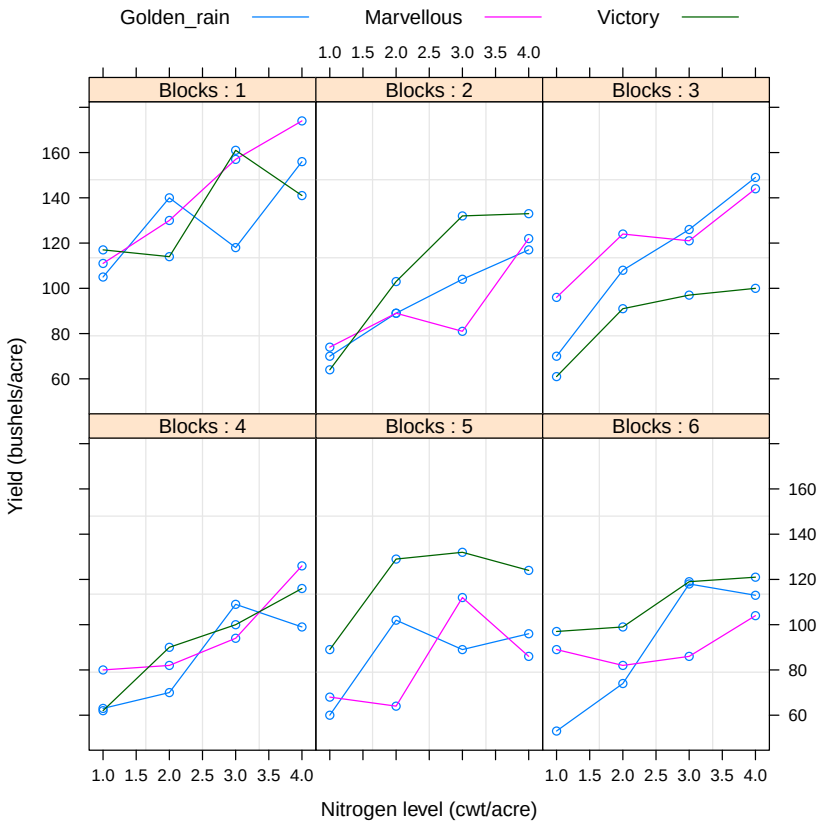

Figure 1.1: Trellis plot of oat yield for each of six blocks.

There is an option in `asreml` to compute approximate stratum variances for linear mixed models in which the variance models for all terms in the random model are simple variance components models. This method, due to Thompson (1980), is exact for orthogonal designs such as this example. The usual ANOVA divides the data into four strata (including the overall mean), namely `Blocks, Blocks:Wplots` and `Blocks:Wplots:Subplots`, with information on the three fixed effects occurring in each of only one of the four strata. That is the main effect of `Variety` occurs in the `Blocks:Wplots` stratum while the main effect of `Nitrogen` and its interaction with `Variety` occur in the `Blocks:Wplots:Subplots` stratum. The REML estimates of the stratum residual variances can be computed from either **wald.asreml** or **svc.asreml**, the latter being the preferred option, as follows:

```
> ss <- svc(oats.vars.asr)
> ss
                          df variance Blocks!Blocks.var Blocks:Wplots!Wplots.var
Blocks!Blocks.var          5 3175.053                12                         4
Blocks:Wplots!Wplots.var  10  601.330                 0                         4
R!units.var               45  177.083                 0                         0
```

```
                          R!units.var
Blocks!Blocks.var                   1
Blocks:Wplots!Wplots.var            1
R!units.var                         1
```

The REML estimates of the stratum variance (displayed in column labelled `variance`) are linear combinations of the REML estimates of the variance components from the fit and the coefficients in the above final three columns of the above matrix, as follows:

```
> ss[,3:5]%*%oats.vars.asr$gammas[-3]
                                [,1]
Blocks!Blocks.var         3175.0556
Blocks:Wplots!Wplots.var   601.3306
R!units.var                177.0833
```

We now turn to producing a summary of the above analysis using the `predict.asreml` method. The mathematics of prediction for linear (mixed) models seems straightforward, although development of a generalised prediction algorithm and method has proven challenging. There are many issues which need to be considered and these have been discussed in detail by Lane & Nelder (1982) for linear and generalised linear models. Welham et al. (2004) extended their ideas for the problem of prediction in linear mixed models, which has the additional complication involving the role of the random effects in forming predictions. Here we will simply illustrate some of the basic ideas necessary for producing meaningful summaries for these data.

`Nitrogen` was the only term to achieve statistical significance at a nominal 5% level and the predicted means for this term can be obtained from the following

```
oats.pvs <- predict(oats.vars.asr,classify='Nitrogen',maxiter=1,trace=FALSE)
```

The object `oats.pvs` is a list with numerous named components, the most relevant being `predictions` which is again a list object containing a data-frame of predictions as well as other relevant information such as standard errors for predicted means for fixed effects or prediction error variances for prediction of random effects or a mizture of both fixed and random effects. The predicted means for the `Nitrogen` treatments are

```
> oats.pvs$predictions
$pvals

Notes:
- The predictions are obtained by averaging across the hypertable
  calculated from model terms constructed solely from factors in
  the averaging and classify sets.
- Use "average" to move ignored factors into the averaging set.

- The SIMPLE averaging set:  Variety
- The ignored set:  Blocks Wplots
```

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

$avsed
 overall
4.435755
```

These predicted means have been formed by averaging of the two-way *hyper* table, which is classified by `Variety` and `Nitrogen`, over the levels of `Variety`. The cells of the two-way table contain the effects of each of the fixed terms in the model, namely `(Intercept, Variety, Nitrogen` and `Variety:Nitrogen`. This is reasonable for these data as the design is orthogonal but in general these predictions would not be meaningful (see, for example (Nelder, 1994) for a lengthy discussion of such issues).

2 Data-sets used in the course

2.1 Experiment and genetic design

In this chapter we present a description of the data-sets which will be used throughout the course. These data-sets have been kindly provided by the RPBC and are a subset of a large multi-environment trial data-set spanning planting dates from 1968 to 2005 and involving 82 experiments.

Table 2.1: Summary information for each experiment

Expt	Plots	Plotsmc	Treesmc	Families	Females	Males	Sets	Iblk	mn
FR203_1	5176	4700	4700	942	189	5	6	1	249
FR203_2	4496	4121	4121	813	165	5	5	1	187
FR203_3	1984	1815	1815	485	98	5	3	1	214
FR353_1	2627	2580	535	42	18	15	1	9	213
FR353_2	2035	1971	524	42	18	15	1	9	180
FR353_3	2403	2300	516	41	17	14	1	9	175

Table 2.1 presents a summary of the six trials. There were two series each with three trials. The first series (ie. FR203) was planted in 1993 and consisted of trees from closed-pollinated (CP) families. The CP families were produced by crossing 5 males with a large number of female trees. The number of trees per cross varied from 1 to 11. The blocking factors included so-called replicates, of which there were 30 and sets within replicates which varied from 3 to 6 between trials. The randomization of trees to replicates and sets within replicates was restricted so that each replicate contained progeny from nearly all female and male parents. However, five replicates were required to accommodate a tree from each family and the same families were randomized to each sub-replicate across replicates (viz groups of 5 sub-replicates). For example, replicates $1, 6, 11, \ldots$ contained one set of families, replicates $2, 7, 12, \ldots$ contained another (disjoint) set of families and so on. This resulted in virtually no family connectivity between sub-replicates within groups of replicates.

The second series of three trials were planted in 1999 and consisted of approximately 530 clonal trees from 42 families. The experiment design was an incomplete block design with 5 replicates and 9 incomplete blocks per replicate. Each incomplete block contained about 60 trees, and between 6-7 families. Each family had 10 clones and clonal trees were replicated five times.

Each clone was planted (nearly so) in each replicate.

All trials contained so-called control families. Most of the controls are mixed parent seedlots in which the individual parent cannot be identified and therefore we have excluded these from the analyses. This can be done in at least two ways, one the straightforward approach achieved by deletion of those records from control trees, the other which fits an effect for each control tree. We will illustrate both approaches as the there are situations where the latter can be useful.

2.2 Connectivity between trials

One of the most important issues in undertaking an analysis of MET data-sets such as these data, is the assessment of the degree of genetic connectivity. Typically, in the analysis of MET data-sets for in-bred crops such as most cereal grain crops, this is measured using the variety (in this case clonal) concurrence matrix. It is rare to have complete balance, in which all varieties or entries are in common across all environments (ie. trials). For trials involving genetically unique individual such as trials using trees from open-pollinated (OP) or CP families, we could also consider a parental concurrence matrix. This is a matrix of order the number of trials whose diagonal elements are the number of parents used in the trial and the off-diagonal elements are the number of parents in common, considering both female and male parents together. This measure has been recently used, for example by Baltunis et al. (2010) and Apiolaza (2012).

We are not aware of any study which delineates sufficient connectivity to permit reliable estimation of genetic parameters, such as genetic correlation between environments, as this would be affected by a range of factors such as the type of variance model used to model Genotype by Environment (G×E) interaction, as well as the heritability of the trait.

Table 2.2 presents the parental connectivity matrix for the six trials used in these notes. There is reasonable connectivity within and between series. Table 2.3 presents the tree connectivity matrix for the three clonal trials. There is excellent connectivity for these three trials at the clonal level.

Table 2.2: Parental connectiviy for the six experiments used in these notes

	FR203_1	FR203_2	FR203_3	FR353_1	FR353_2	FR353_3
FR203_1	194	170	103	21	21	20
FR203_2	170	170	79	18	18	17
FR203_3	103	79	103	13	13	13
FR353_1	21	18	13	24	24	23
FR353_2	21	18	13	24	24	23
FR353_3	20	17	13	23	23	23

2.3 A brief review & tutorial

Before considering the information on pedigrees for these trials we will now revisit the R commands which we used to read in the data, and undertake various, standard checks which we have

Table 2.3: Clonal connectivity for the three clonal experiments used in these notes

	FR353_1	FR353_2	FR353_3
FR353_1	535	517	516
FR353_2	517	524	515
FR353_3	516	515	516

found useful in our treatment of such data-sets. We begin by sourcing in the various functions and other useful packages such as **ASReml-R**. We usually do this as a default. Note that there are other automatic ways to do this in R which we will cover later in the course. The code to read in the data, delete the first two columns, order the data frame (`ws.df`) and define the parental control trees is as follows

```
> require(asreml)
> source("C:/research/Rscripts/pedtrim.R")
> source("C:/research/Rscripts/nfa-fns.R")
> ws.df <- read.table("c:/courses/vancouver/scripts/wsdata.csv",header=T,sep=',')
> ws.df <- ws.df[,-c(1,2)]
> ws.df <- ws.df[order(ws.df$Expt),]
> control.numbers <- c(111020,111039,111111,111227,111228,111266,111268,111270,111525,
+                      111526,
+                      111527,111602,111606,111607,111613,111614,111616,111618,111619,
+                      111626,
+                      111628,111630,111646,111647,111652,111674,111676,111678,111713,
+                      111791,
+                      111792,111854,111870,111871,111872,111930,111991,111992,111993,
+                      111997,
+                      111998)
```

Note that we *always* recommend annotating scripts with comments, but these have been removed in these notes for ease of presentation. Once the data has been read into R, we now turn to determining some basic statistics and properties of the data-set. Two useful commands for examining the data-frame are given below. Note that we have only printed the first 8 columns of `ws.df` for ease of presentation:

```
> head(ws.df[,1:8])

     Expt nws Plantdate Plottype Cp Set Block Clone
1 FR203_1   1      1993        1  2   2     1     1
2 FR203_1   2      1993        1  2   2     1     1
3 FR203_1   3      1993        1  2   2     1     1
4 FR203_1   4      1993        1  2   2     1     1
5 FR203_1   5      1993        1  2   2     1     1
6 FR203_1   6      1993        1  2   2     1     1

> str(ws.df)
```

```
'data.frame':        18721 obs. of  24 variables:
 $ Expt     : Factor w/ 6 levels "FR203_1","FR203_2",..: 1 1 1 1 1 1 1 1 1 ...
 $ nws      : int  1 2 3 4 5 6 7 8 9 10 ...
 $ Plantdate: int  1993 1993 1993 1993 1993 1993 1993 1993 1993 1993 ...
 $ Plottype : int  1 1 1 1 1 1 1 1 1 ...
 $ Cp       : int  2 2 2 2 2 2 2 2 2 ...
 $ Set      : int  2 2 2 2 2 2 2 2 2 ...
 $ Block    : int  1 1 1 1 1 1 1 1 1 ...
 $ Clone    : int  1 1 1 1 1 1 1 1 1 ...
 $ age      : int  8 8 8 8 8 8 8 8 8 ...
 $ Replicate: int  1 1 1 1 1 1 1 1 1 ...
 $ Setgroup : Factor w/ 7 levels "0","A","B","C",..: 2 2 2 2 2 2 2 2 2 ...
 $ Treeno   : int  1 2 3 4 5 6 7 9 10 11 ...
 $ Plot     : int  0 0 0 0 0 0 0 0 0 ...
 $ Siteplot : int  0 0 0 0 0 0 0 0 0 ...
 $ Iblk     : int  0 0 0 0 0 0 0 0 0 ...
 $ Tree     : int  43000001 43000002 43000003 43000004 43000005 43000006 43000007 43000009 43000010 4
 $ Tdio     : int  43000001 43000002 43000003 43000004 43000005 43000006 43000007 43000009 43000010 4
 $ Fcln     : int  850227 850112 111791 850055 111614 188305 268109 850272 111713 268315 ...
 $ Mcln     : int  875043 875043 NA 875043 NA 875043 875043 875043 NA 875043 ...
 $ Family   : int  1696 1655 0 1544 0 1106 1151 1726 0 1196 ...
 $ br9      : int  7 4 3 8 4 4 6 6 8 6 ...
 $ dbh      : num  229 282 217 251 238 283 267 258 253 282 ...
 $ str      : int  6 3 3 7 6 6 7 7 8 4 ...
 $ Trueplot : Factor w/ 185 levels "1.0.0","1.A.0",..: 2 2 2 2 2 2 2 2 2 ...
```

The `str()` function returns a summary of the columns of the data-frame used as the first argument. By default `read.table` converts columns which contain characters to `factors`, but this can be controlled by the `as.is` argument.

The following commands determine some of the basic parental information which is usually an important initial step in undertaking a genetic analysis using **ASReml-R**.

```
> female.names <- unique(ws.df$Fcln)
> male.names <- unique(ws.df$Mcln)
> male.names <- male.names[!is.na(male.names)]
> parent.names <- c(female.names,male.names)
> both.names <- parent.names[duplicated(parent.names)]
> all.parent.names <- parent.names
> table(table(all.parent.names))

  1   2
191  14

> controls.used <- parent.names[is.element(parent.names,control.numbers)]
> parent.names <- parent.names[!is.element(parent.names,control.numbers)]
> parent.names <- unique(parent.names)
> c(length(both.names),length(female.names),length(male.names),
+ length(parent.names),length(controls.used))
```

```
[1]  14 199  20 197   8
```

and we see that, for example, 8 control parents were used, 199 female parents were used, 20 male parents were used and 197 had progeny in these trials.

A reduced data-frame excluding control trees is easily formed using

```
> wsmc.df <- subset(ws.df,!is.element(Fcln,control.numbers))
```

3 Analysis of individual tree breeding experiments

3.1 Introduction

In this chapter we present an overview of an approach to the analysis of individual tree breeding experiments, where we aim to estimate genetic variance parameters and predict genetic effects, focussing mainly on prediction of so-called additive breeding values. We do not present details on the underlying genetic models and theory; the reader is referred to texts such as Mrode (1995); Lynch & Walsh (1998).

ASReml-R has a range of facilities for undertaking genetic analyses, in particular fitting models where information on pedigrees is incorporated into the analysis. However, to be able to choose the appropriate terms to include in the analyses it is important to understand the process which determines the composition of the linear mixed model, and what determines whether a term is *fixed, random* and whether it should always be included in the model. In section 3.2 we present a linear mixed model which provides the basis for the analysis of the data-sets presented in chapter 2 and can be easily extended to the analysis of MET data-sets and/or the analysis of multiple traits. This model is a special case of equation 1.1.1 but we include more details on the form of the variance models which are generally applied to each model term. In section 3.2.1 we then present the so-called reduced animal (tree) model which we have found to be extremely useful in the analysis of large MET data-sets, as it overcomes some of the computational hurdles in fitting complex variance models to these types of data.

3.2 Statistical Methods

For generality we begin by considering the analysis of the j^{th} trial ($j = 1 \dots t$) within the series of trials described in chapter 2. We have referred to these data as a multi-environment trial (MET) data-set. We assume that the j^{th} trial comprises n_j plots (to be pedantic, observational units). In the analysis of field trials for cereal crops, say, it is usual that the spatial co-ordinates of the plots (ie trees in this case) are readily available. This then allows use of the more efficient spatial analysis techniques (for example those used by Stefanova et al. (2009)). The spatial co-ordinates of the trees in these trials were not readily available and so the traditional complete and incomplete block models have been used for modelling the non-genetic variation within a

trial.

The statistical model for $\boldsymbol{y}_j$, the vector of data for trial j, can be written as

$$\boldsymbol{y}_j = \boldsymbol{X}_j\boldsymbol{\tau}_j + \boldsymbol{Z}_{g_j}\boldsymbol{u}_{g_j} + \boldsymbol{Z}_{p_j}\boldsymbol{u}_{p_j} + \boldsymbol{e}_j \tag{3.2.1}$$

where $\boldsymbol{\tau}_j$ is a vector of fixed effects with associated design matrix $\boldsymbol{X}_j$ (assumed to have full column rank); $\boldsymbol{u}_{g_j}$ is the $m \times 1$ vector of random genetic effects with associated design matrix $\boldsymbol{Z}_{g_j}$; $\boldsymbol{u}_{p_j}$ is a vector of random non-genetic (or peripheral) effects with associated design matrix $\boldsymbol{Z}_{p_j}$ and $\boldsymbol{e}_j$ is the vector of residuals for the trial. In the simplest case the vector $\boldsymbol{\tau}_j$ comprises an overall mean (intercept) for the trial but may include other effects as necessary. The vector $\boldsymbol{u}_{p_j}$ includes effects for blocking factors associated with the trial design. The composition of this vector varies between trials, depending on the design (ie. complete or incomplete block design), the use of sets within replicates and multi-tree plots.

For the remainder of this section, we will drop the j suffix in order to avoid cluttered notation. This suffix will be reintroduced when we consider the analysis of MET data-sets in later chapters. Thus equation 3.2.1 becomes

$$\boldsymbol{y} = \boldsymbol{X}\boldsymbol{\tau} + \boldsymbol{Z}_g\boldsymbol{u}_g + \boldsymbol{Z}_p\boldsymbol{u}_p + \boldsymbol{e} \tag{3.2.2}$$

We assume that the $\boldsymbol{u}_g, \boldsymbol{u}_p$ and $\boldsymbol{e}$ vectors of random effects are mutually independent, and distributed as multivariate Gaussian, with zero means. The variance matrices of the vectors of non-genetic effects and residual errors are given by $\boldsymbol{G}_p = \oplus_{k=1}^{b}\sigma^2_{p_k}\boldsymbol{I}_{q_k}$ and $\sigma^2\boldsymbol{I}_n$ respectively where $b = b'$ (see section 1.1.3) is the number of components in $\boldsymbol{u}_p$ and q_k is the number of effects in (length of) $\boldsymbol{u}_{p_k}$.

We consider a simple model for $\boldsymbol{u}_g$ given by

$$\boldsymbol{u}_g = \boldsymbol{u}_a + \boldsymbol{u}_e$$

where the two terms represent the additive and non-additive (or residual) genetic effects. More complex models including ones which partition the non-additive into dominance and residual genetic (ie. non-additive and non-dominance) can also be considered if necessary. These vectors, namely $\boldsymbol{u}_s, s = g, a, e$ are partitioned into two sub-vectors, the first representing those trees with progeny, and the latter representing those trees without progeny. This partition is denoted by $\boldsymbol{u}_s^\intercal = (\boldsymbol{u}_{s_p}^\intercal,\ \boldsymbol{u}_{s_o}^\intercal), s = g, a, e$. There is a conformal partitioning of the columns of $\boldsymbol{Z}_g$ given by $\boldsymbol{Z}_g = [\boldsymbol{Z}_{g_p}\ \boldsymbol{Z}_{g_o}]$. The length of $\boldsymbol{u}_{s_p}$ is m_p and typically m_p is substantially less than m. For example, in the full RPBC MET data-set there are about 290,000 unique trees in the full MET data-set, with only 2697 of these having progeny. We note that typically trees with progeny (ie. parental trees) do not occur in the data-set and so $\boldsymbol{Z}_{g_p} = \boldsymbol{0}$. We revisit this later in this chapter.

We assume that the variance matrices of the vectors of additive and non-additive genetic effects are given by

$$\begin{aligned} \text{var}\,(\boldsymbol{u}_a) &= \sigma^2_a\boldsymbol{A} \quad \text{and} \\ \text{var}\,(\boldsymbol{u}_e) &= \sigma^2_e\boldsymbol{I}_m \end{aligned}$$

where $\boldsymbol{A}$ is the numerator relationship matrix with partitioning conformal with $\boldsymbol{u}_a$.

It follows that the distribution of $\boldsymbol{y}$ for non-clonal trials (ie. trials which contain trees that are progeny of OP or CP families, having no data on parental trees) is multivariate Gaussian with

$$\begin{aligned} \mathrm{E}(\boldsymbol{y}) &= \boldsymbol{X}\boldsymbol{\tau} \\ \mathrm{var}(\boldsymbol{y}) &= \sigma_a^2 \boldsymbol{Z}_g \boldsymbol{A} \boldsymbol{Z}_g^{\intercal} + \boldsymbol{Z}_p \boldsymbol{G}_p \boldsymbol{Z}_p^{\intercal} + \sigma_e^{2*} \boldsymbol{I}_n \end{aligned} \tag{3.2.3}$$

where $\sigma_e^{2*} = \sigma_e^2 + \sigma^2$.

For clonal trials (assuming no data on parental trees and removing data on control trees) the distribution of $\boldsymbol{y}$ is multivariate Gaussian with

$$\begin{aligned} \mathrm{E}(\boldsymbol{y}) &= \boldsymbol{X}\boldsymbol{\tau} \\ \mathrm{var}(\boldsymbol{y}) &= \boldsymbol{Z}_g(\sigma_a^2 \boldsymbol{A} + \sigma_e^2 \boldsymbol{I}_m)\boldsymbol{Z}_g^{\intercal} + \boldsymbol{Z}_p \boldsymbol{G}_p \boldsymbol{Z}_p^{\intercal} + \sigma^2 \boldsymbol{I}_n \end{aligned} \tag{3.2.4}$$

3.2.1 Reduced Animal (RA) model for single trial analysis

Background

The above model for the analysis of single trials is often referred to as the (full) animal model. There can be substantial computing cost associated with fitting this so-called animal model. This burden is not substantial for the analysis of individual trials, as **ASReml-R** uses sparse matrix methods and efficient equation ordering to maintain sparsity in the mixed model equations which renders computation of the inverse coefficient matrix feasible, even with greater than $250,000$ individuals. Of course, separate pedigree files can be created and used for each trial, which then reduces the dimension of the coefficient matrix in the mixed model equations, but this approach is clumsy and error prone. The reduction in computing time using separate pedigree files is not sufficient to warrant use of this approach.

We also note that the aim of our analysis is to undertake a full one-stage MET analysis which involves exploring (additive) genetic by environment interaction. This analysis can often involve estimation of complex variance structures, such as unstructured and so-called factor analytic models, in order to model the variance structure of the additive (and non-additive) genetic by environment interaction effects (see for example Cullis et al. (2010)). This analysis would then require formation of a set of mixed model equations of the order of $21e6$. This would be computationally prohibitive even for the most simplistic of variance models.

We can significantly reduce the dimension of the mixed model equations by use of the so-called reduced animal (RA) model. The RA model was first introduced by Quass & Pollack (1980) to obtain predicted additive genetic effects (ie. breeding values) without the need to set up equations for all individuals. With the advent of efficient computing software and hardware platforms this model has lost favour. Most breeders believe that the computational benefits of the RA model are outweighed by the difficulty of setting it up. Furthermore, the full animal model is normally used for the estimation of variance components and breeding values are

obtained as a by-product of the estimation algorithm for most software, including **ASReml-R**. White et al. (2006) showed how the RA model can be used to both estimate variance components and predict breeding values using standard software. Their formulation exploits the fact that the portion of the inverse of the numerator relationship matrix associated with the non-parental lines is diagonal. An additional term can be then included in the model, fitted as a random effect, with known weights, and variance parameters constrained to be equal to the additive variance component to achieve full REML estimation. An approximate approach would be to ignore this additional term and inbreeding. In the following we present a brief derivation of the RA model and argue that the approximate model provides sufficiently accurate estimates of the variance components. The approximate RA model has further appeal, since the aim of our analysis is primarily to predict additive genetic values for only parental lines (ie. backward selections), but can be simply modified to include prediction of breeding values for forward selections which is relevant to clonal trials. This modification is presented in the following section.

RA model

We denote the numerator relationship matrix by

$$\boldsymbol{A} = \begin{bmatrix} \boldsymbol{A}_{pp} & \boldsymbol{A}_{po} \\ \boldsymbol{A}_{op} & \boldsymbol{A}_{oo} \end{bmatrix}$$

and note that

$$\boldsymbol{u}_{a_o} = \boldsymbol{T}_{op}\boldsymbol{u}_{a_p} + \boldsymbol{u}_{m_o} \tag{3.2.5}$$

where $\boldsymbol{u}_{m_o}$ represents the so-called Mendelian variation for the non-parental trees, and the matrix $\boldsymbol{T}_{op}$ is a parent indicator matrix given by

$$\boldsymbol{T}_{op} = \frac{1}{2}(\boldsymbol{F}_{op} + \boldsymbol{M}_{op})$$

where $\boldsymbol{F}_{op}$ and $\boldsymbol{M}_{op}$ are female and male parent indicator matrices respectively. It follows that

$$\boldsymbol{A} = \begin{bmatrix} \boldsymbol{A}_{pp} & \boldsymbol{A}_{pp}\boldsymbol{T}_{op}^{\intercal} \\ \boldsymbol{T}_{op}\boldsymbol{A}_{pp} & \boldsymbol{T}_{op}\boldsymbol{A}_{pp}\boldsymbol{T}_{op}^{\intercal} + \boldsymbol{D}_{oo} \end{bmatrix}$$

where $\text{var}(\boldsymbol{u}_{m_o}) = \sigma_a^2\boldsymbol{D}_{oo}$ is a diagonal matrix, with elements given by $\frac{1}{2}(1 - d_{ii})$, d_{ii} being the inbreeding coefficient for the ith non-parental individual (tree). Using the properties of the inverse of a partitioned matrix it follows that for $\boldsymbol{Q}_{oo} = \boldsymbol{D}_{oo}^{-1}$

$$\boldsymbol{A}^{-1} = \begin{bmatrix} \boldsymbol{A}_{pp}^{-1} + \boldsymbol{T}_{op}^{\intercal}\boldsymbol{Q}_{oo}\boldsymbol{T}_{op} & -\boldsymbol{T}_{op}^{\intercal}\boldsymbol{Q}_{oo} \\ -\boldsymbol{Q}_{oo}\boldsymbol{T}_{op} & \boldsymbol{Q}_{oo} \end{bmatrix}$$

Substituting equation 3.2.5 into equation 3.2.2 gives

$$\boldsymbol{y} = \boldsymbol{X}\boldsymbol{\tau} + (\boldsymbol{Z}_{g_p} + \boldsymbol{Z}_{g_o}\boldsymbol{T}_{op})\boldsymbol{u}_{a_p} + \boldsymbol{Z}_{g_o}\boldsymbol{u}_{m_o} + \boldsymbol{Z}_g\boldsymbol{u}_e + \boldsymbol{Z}_p\boldsymbol{u}_p + \boldsymbol{e} \tag{3.2.6}$$

This is the RA model. In most of our applications we do not have data on parental individuals, however for complete generality, and for the prediction of breeding values for forward selections,

we now consider a further partition of the data vector into two components. For ease of presentation, and without loss of generality, we assume that the data vector $\boldsymbol{y} = (\boldsymbol{y}_p^\intercal, \boldsymbol{y}_o^\intercal)^\intercal$ say, where each component is a vector of length n_p or n_o respectively and $n = n_p + n_o$. Using this partition the design matrix for $\boldsymbol{u}_{a_p}$ (ie. the vector of additive genetic values for the parental trees) is given by

$$\begin{aligned} \boldsymbol{Z}_{g_p} + \boldsymbol{Z}_{g_o}\boldsymbol{T}_{op} &= \begin{bmatrix} \boldsymbol{Z}_{g_{pp}} \\ \boldsymbol{0} \end{bmatrix} + \begin{bmatrix} \boldsymbol{0} \\ \boldsymbol{Z}_{g_{oo}}\boldsymbol{T}_{op} \end{bmatrix} \\ &= \begin{bmatrix} \boldsymbol{Z}_{g_{pp}} \\ \boldsymbol{0} \end{bmatrix} + \begin{bmatrix} \boldsymbol{0} \\ \frac{1}{2}(\boldsymbol{Z}_{f_{op}} + \boldsymbol{Z}_{m_{op}}) \end{bmatrix} \end{aligned}$$

where $\boldsymbol{Z}_{f_{op}} = \boldsymbol{Z}_{g_{oo}}\boldsymbol{F}_{op}$ and $\boldsymbol{Z}_{m_{op}} = \boldsymbol{Z}_{g_{oo}}\boldsymbol{M}_{op}$ are the design matrices for females and males for the non-parental trees. Hence it follows that

$$\begin{aligned} \boldsymbol{Z}_{g_p} + \boldsymbol{Z}_{g_o}\boldsymbol{T}_{op} &= \begin{bmatrix} \frac{1}{2}(\boldsymbol{Z}_{g_{pp}} + \boldsymbol{Z}_{g_{pp}}) \\ \frac{1}{2}(\boldsymbol{Z}_{f_{op}} + \boldsymbol{Z}_{m_{op}}) \end{bmatrix} \\ &= \frac{1}{2}(\boldsymbol{Z}_f + \boldsymbol{Z}_m) \end{aligned}$$

where

$$\boldsymbol{Z}_f = \begin{bmatrix} \boldsymbol{Z}_{g_{pp}} \\ \boldsymbol{Z}_{f_{op}} \end{bmatrix} \qquad \text{and} \qquad \boldsymbol{Z}_m = \begin{bmatrix} \boldsymbol{Z}_{g_{pp}} \\ \boldsymbol{Z}_{m_{op}} \end{bmatrix}$$

are the design matrices for female and male parents, where the female and male parents are coded as themselves for parental trees.

Although for our examples there is usually no data on parental trees, this result provides the framework for coding to allow prediction of breeding values for forward selection in clonal trials. The additional feature of the above design matrix is that we have to have the ability to overlay two design matrices (or in general form a linear combination of design matrices). **ASReml-R** can do the former using the `and` facility within the model formula syntax. This facility will also be demonstrated in the following chapters.

For our data-sets we have two situations to consider in detail, trials which contain trees which are progeny of either OP or CP families and trees which are clonal trees from (generally) CP crosses. For the former trial type, we have

- $\boldsymbol{y} = \boldsymbol{y}_o$ and $\boldsymbol{Z}_{g_p} = \boldsymbol{0}$, since there is no data on parental trees, and
- $\boldsymbol{Z}_{g_{oo}} = \boldsymbol{I}_n = \boldsymbol{I}_{m_o}$.

Hence

$$\boldsymbol{Z}_{g_p} + \boldsymbol{Z}_{g_o}\boldsymbol{T}_{op} = \boldsymbol{T}_{op} = \frac{1}{2}(\boldsymbol{F}_{op} + \boldsymbol{M}_{op})$$

Using this result equation 3.2.6 becomes

$$\begin{aligned} \boldsymbol{y} &= \boldsymbol{X}\boldsymbol{\tau} + \frac{1}{2}(\boldsymbol{F}_{op} + \boldsymbol{M}_{op})\boldsymbol{u}_{a_p} + \boldsymbol{u}_{m_o} + \boldsymbol{u}_{e_o} + \boldsymbol{Z}_p\boldsymbol{u}_p + \boldsymbol{e} \\ &= \boldsymbol{X}\boldsymbol{\tau} + \frac{1}{2}(\boldsymbol{F}_{op} + \boldsymbol{M}_{op})\boldsymbol{u}_{a_p} + \boldsymbol{Z}_p\boldsymbol{u}_p + \boldsymbol{e}_a^* \end{aligned} \tag{3.2.7}$$

where $\boldsymbol{e}_a^*$ is a composite residual with variance matrix given by $\sigma_a^2\boldsymbol{D}_{oo}+\sigma_e^2\boldsymbol{I}_n+\sigma^2\boldsymbol{I}_n$. Following White et al. (2006) ignoring inbreeding then the approximate variance matrix is $(\frac{1}{2}\sigma_a^2+\sigma_e^2+\sigma^2)\boldsymbol{I}_n$.

For clonal trials, again $\boldsymbol{y}=\boldsymbol{y}_o$, and $\boldsymbol{Z}_{g_p}=\boldsymbol{0}$, but $\boldsymbol{Z}_{g_{oo}}\neq\boldsymbol{I}_n$ so that

$$\boldsymbol{Z}_{g_p}+\boldsymbol{Z}_{g_o}\boldsymbol{T}_{op}=\boldsymbol{Z}_{g_{oo}}\boldsymbol{T}_{op}=\frac{1}{2}(\boldsymbol{Z}_{f_{op}}+\boldsymbol{Z}_{m_{op}})$$

Using this result equation 3.2.6 becomes

$$\begin{aligned}\boldsymbol{y} &= \boldsymbol{X}\boldsymbol{\tau}+\boldsymbol{Z}_{g_{oo}}\boldsymbol{T}_{op}\boldsymbol{u}_{a_p}+\boldsymbol{Z}_{g_{oo}}\boldsymbol{u}_{m_o}+\boldsymbol{Z}_{g_{oo}}\boldsymbol{u}_{e_o}+\boldsymbol{Z}_p\boldsymbol{u}_p+\boldsymbol{e}\\ &= \boldsymbol{X}\boldsymbol{\tau}+\frac{1}{2}(\boldsymbol{Z}_{f_{op}}+\boldsymbol{Z}_{m_{op}})\boldsymbol{u}_{a_p}+\boldsymbol{Z}_{g_{oo}}\boldsymbol{u}_{a_o}^*+\boldsymbol{Z}_p\boldsymbol{u}_p+\boldsymbol{e}\end{aligned}\tag{3.2.8}$$

where $\boldsymbol{u}_{a_o}^*$ is a composite genetic residual comprising Mendelian sampling and residual genetic effects, with variance matrix given by $\sigma_a^2\boldsymbol{D}_{oo}+\sigma_e^2\boldsymbol{I}_{m_o}$. Again, ignoring inbreeding, the approximate variance matrix is $(\frac{1}{2}\sigma_a^2+\sigma_e^2)\boldsymbol{I}_{m_o}$.

4 Analysis of individual tree breeding experiments: Examples

4.1 Introduction

In this chapter we present the analysis of two individual tree breeding experiments, taken from the six trials described in chapter 2. The two trials we consider in detail are FR203_1 (a trial consisting of individuals from open-pollinated families) and FR353_1 (a trial consisting of cloned individuals from closed-pollinated families). We denote these trials as *OP* and *Clonal* trials. We will illustrate how to fit the so-called tree, reduced animal and approximate reduced animal models in **ASReml-R**. We will also present the approach for computing reliabilities after Mrode (1995) as well as obtaining forward selections for Clonal trials.

4.2 Example 1: Analysis of an OP trial

4.2.1 Preliminaries: data construction

In this section we will present a comprehensive analysis of an OP trial using the models presented in chapter 3. The notes will closely follow the associated script file (`ssop.R`) which has been distributed with the notes and data-sets on the USB stick. We have taken a standard approach in developing these scripts and included the pre-processing R commands which read in the data and set-up the basic data frame and structures which are necessary for the script file to run without dependence on other script files. In practice, it would be unnecessary to include these commands in all scripts. However for teaching and demonstration purposes we feel that it avoids unnecessary confusion and reduces the chances for downstream errors to occur.

The header material which we refer to is given below.

```
> ##################
> # this is the script to run the single site OP analyses
> # we will read in the data and ped files and then
> # conduct tree, RAM and  ARAM analyses
> #
> ######################
> require(asreml)
```

```
> source("C:/courses/vancouver/scripts/pedtrim.R")
> source("C:/courses/vancouver/scripts/nfa-fns.R")
> ###############
> # first read in the data and ped files
> #
> # set up control trees
> # 8 control trees in pedigree and data
> ################
> ws.df <- read.table("c:/courses/vancouver/scripts/wsdata.csv",
+                     header=T,sep=',')
> ws.df <- ws.df[,-c(1,2)]
> ws.df <- ws.df[order(ws.df$Expt),]
> wsped.df <- read.table("c:/courses/vancouver/scripts/wspedtree.csv",
+                        header=T,sep=',')
> wsped.df <- wsped.df[,-1]
> head(wsped.df)
> nrow(wsped.df) # 12714
> control.numbers <- c(111020,111039,111111,111227,111228,111266,111268,
+                      111270,111525,111526,
+                      111527,111602,111606,111607,111613,111614,111616,
+                      111618,111619,111626,
+                      111628,111630,111646,111647,111652,111674,111676,
+                      111678,111713,111791,
+                      111792,111854,111870,111871,111872,111930,111991,
+                      111992,111993,111997,111998)
> nrow(subset(wsped.df,is.element(Tree,control.numbers))) #8
```

In this section of the script file we read in functions which we (may) use in the script (these functions are also supplied with the notes on the USB stick). As well, we read in the data and pedigree files and create the object, `control.numbers`, which defines the parental trees which are so-called *control trees*.

In the next section of the script file we create three data frames, one for each of the three OP trials. We will only consider the analysis of the first trial (ie. FR203_1, or `op1.df`). We then use a supplied function to create pedigree data frames for individuals and parental trees for each of the three trials. As mentioned in chapter 2 this can reduce some of the computational load, and being useful for fitting the reduced animal and approximate reduced animal (ARA) model.

```
> #################
> # select the three op sites
> # and use pedtrim to reduce pedigrees to be specific to each trial
> # 12714 in full pedigree for 6 trials
> #
> #################
> op.sites <- c('FR203_1','FR203_2', 'FR203_3')
> op1.df <- subset(ws.df,is.element(Expt,op.sites[1]))
> op2.df <- subset(ws.df,is.element(Expt,op.sites[2]))
> op3.df <- subset(ws.df,is.element(Expt,op.sites[3]))
```

```
> temp.ped <- data.ped.trim(pedfile=wsped.df,data=op1.df,parent=TRUE)
> op1.tree.df <- temp.ped$tree.ped #5472
> op1.parent.df <- temp.ped$parent.ped #296
> temp.ped <- data.ped.trim(pedfile=wsped.df,data=op2.df,parent=TRUE)
> op2.tree.df <- temp.ped$tree.ped #4766
> op2.parent.df <- temp.ped$parent.ped #270
> temp.ped <- data.ped.trim(pedfile=wsped.df,data=op3.df,parent=TRUE)
> op3.tree.df <- temp.ped$tree.ped #2158
> op3.parent.df <- temp.ped$parent.ped #174
```

There are 5472 trees in the full pedigree data frame for `op1.df` compared to only 296 individuals in the parental data frame.

```
> ####################
> # Set up the Check factor
> # as I said in the course notes you can drop the data records for the Checks
> # or leave them in and fit the Check factor as a fixed effect
> # the end result is the same
> # the latter approach is a neat trick which we choose to demonstrate
> # set up the control factor which is based on Fcln and Mcln numbers
> # >111000 and <111999
> # this creates a factor with 477 levels = 476 + 1
> ####################
> temp <- rep(1,nrow(op1.df))
> temp[is.element(op1.df$Fcln,control.numbers)] <-
+     op1.df$Tree[is.element(op1.df$Fcln,control.numbers)]
> length(unique(temp))

[1] 477

> temp[is.element(op1.df$Mcln,control.numbers)] <-
+     op1.df$Tree[is.element(op1.df$Mcln,control.numbers)]
> t1 <- subset(op1.df,is.element(op1.df$Mcln,control.numbers))$Tree #0
> t2 <- subset(op1.df,is.element(op1.df$Fcln,control.numbers))$Tree #476
> c(sum(is.element(t1,t2)),sum(is.element(t2,t1)))

[1] 0 0

> c(length(unique(temp)),length(t1)+length(t2))

[1] 477 476

> op1.df$Check <- factor(temp)
> c(nrow(subset(op1.df,is.element(op1.df$Fcln,control.numbers)))+
+   nrow(subset(op1.df,is.element(op1.df$Mcln,control.numbers))),
+   length(levels(op1.df$Check))) # [1] 476 477

[1] 476 477
```

There are 477 levels in the factor `Check`, which is equal to one more than the number of trees which are progeny of control trees. When we fit this factor in the model, all of the data on these trees will effectively be discarded from the analysis. We could also simply create a new data frame without these data records. This is often easier to understand, but may not be the most efficient approach. This is particularly true for the analysis of field trials using spatial models, in which we exploit the separability assumption afforded by the contiguous and regular spatial layout often seen in these trials.

4.2.2 Forming pedigrees and related objects for the RA and ARA models

In this section we form the objects and additional columns within the data frame which are used in the subsequent analyses. This portion of the script file demonstrates the use of the `asreml.Ainverse()` which is a function distributed within the **ASReml-R** package. In any genetic analysis, such as this example, we have phenotypic data on a set of individuals that are genetically linked via a pedigree and typically we wish to include this information in the analysis. We proceed by first creating the additive relationship matrix (ie. $\boldsymbol{A}$), or more specifically the inverse of the additive relationship matrix (ie. $\boldsymbol{A}^{-1}$) as this is what is required by `asreml`. The inclusion of the $\boldsymbol{A}^{-1}$ matrix in the analysis is essentially a two step process:

1. the function `asreml.Ainverse()` takes a pedigree data frame and returns the $\boldsymbol{A}^{-1}$ matrix in sparse form as a `giv` object (see step 2).

2. The matrix from step 1 (giv object) is included in an analysis using the `ginverse` argument in conjunction with the `ped()` variance model function.

For the more general situation, where the pedigree based inverse relationship matrix generated by `asreml.Ainverse()` is not appropriate, the user can include a general inverse (or non-inverse) known variance matrix provided its structure adheres to one of the allowable forms for a giv object. This is how we may include a relationship matrix computed from genomic data. For more details see Butler et al. (2009b, chapter 5).

There may be more than one G-inverse matrix present and each of these matrices is supplied through named components of the `ginverse` argument. The `ped()` and `giv()` special functions in the random model formula associate the appropriate G-inverse with the nominated model factor. We will illustrate this in the following section.

The `asreml.Ainverse()` function computes the inverse relationship matrix from a pedigree data frame supplied through the `pedigree` argument using the methods of Meuwissen & Luo (1992). The pedigree data frame contains (at least) three columns that correspond to the individual, male and female parents respectively. The row giving the pedigree of an individual must appear before any row where that individual appears as a parent. The function attempts to resolve any such omissions, and inserts missing parental records, but defaults to non-inbred individuals with unknown male and female parents. These omissions are reported when the function is called, and it is recommended that this report is carefully examined before proceeding with any

analysis. The function `asreml.Ainverse()` has additional arguments which are not presented here. For a full description see (Butler et al., 2009b, page 78).

The portion of the script to produce the sparse form of the inverse relationship matrices for both the full and parental pedigrees is presented below.

```
> parents.op1 <- op1.parent.df$Tree
> nonparents.op1 <- op1.tree.df$Tree
> nonparents.op1 <- nonparents.op1[!is.element(nonparents.op1,parents.op1)]
> c(length(parents.op1),length(nonparents.op1))

[1]  296 5176

> temp <- asreml.Ainverse(op1.parent.df)
> table(temp$inbreeding)

    0 0.125   0.5
  291     1     4

> op1.parents.ainv <- temp$ginv
> temp <- asreml.Ainverse(op1.tree.df)
> table(temp$inbreeding)

     0 0.0625  0.125   0.25    0.5
  5451      5      7      5      4

> op1.ainv <- temp$ginv
```

Note that there is negligible inbreeding in these pedigrees.

We now compute the elements of $\boldsymbol{D}_{oo}$, which we will use in the RA model. This portion of the script file is for an advanced user of `asreml()`, but we include this for completeness and for illustration of the equivalence of the analyses based on the full tree and RA model, if appropriate terms with constraints are included.

```
> ###################
> # get the D matrix which is the inverse of the diagonal of A^{-1}
> # for those nonparents
> #
> #
> ####################
> temp <- subset(op1.ainv,Row==Column)
> temp$Tree <- as.character(attr(op1.ainv,'rowNames'))
> temp <- subset(temp,is.element(Tree,nonparents.op1))
> temp <- merge(op1.df,temp[,c('Tree','Ainverse')],by='Tree')
> ##########################
> # ok this all checks out
> # the ones with missing Mcln are all Check!=1
```

```
> # and are qii = 1/dii = 1/(.75 - 0) = 1.3333
> # others range from 2
> # which makes sense as qii = 1/dii = 1/(0.5 - 0.25(Fs + Fd))
> # all good now create the sqrt variance term
> #
> ##########################
> op1.df <- temp
> table(with(op1.df,tapply(Ainverse,as.character(Tree),mean)))

1.33333333333333               2 2.13333333333333 2.66666666666667
             476            4573              24             103

> op1.df$vv <- 1/sqrt(op1.df$Ainverse)
```

The additional column `vv` contains the square root of the diagonal elements of $\boldsymbol{D}_{oo}$ with values mostly given by $\sqrt{2}$.

4.2.3 Analyses for the full tree, RA and ARA models

It remains to now undertake the analyses using `asreml()`. The script file contains additional lines of code factorising all variables which may be used in the subsequent analyses as factors. We do not present this for brevity.

We now fit the tree model, noting that the trial was designed using sets within replicates, hence the inclusion of the nested effect of `Setgroup` within `Replicate`. The `ginverse` argument associates the `op1.ainv` object with the `ped(Tree)` model term. Note also that we include `Check` as a fixed effect, nominating it in the `sparse` model formula. This facility is useful when fitting terms as fixed which have a large number of effects, but we are not interested in inference (eg. Wald tests) for such terms. The argument `workspace` is not often used, but needs to be set when fitting models which require larger amounts of computer memory. This model would fit with the default setting for `workspace`, but we retain it in the call to illustrate its use.

```
> op1.tree.asr <- asreml(dbh~1,random=~ped(Tree) +
+                        Replicate + Replicate:Setgroup,
+                        rcov=~units,data=op1.df,ginverse=list(Tree=op1.ainv),
+                        na.method.X='include',
+                        workspace=80e6,sparse=~Check)

asreml 3.0-1 (24 October 2012), Library: 3.0hj (15 November 2011), X86_64
     LogLik         S2      DF      wall     cpu
 -18135.5746    729.6107  4699  16:36:16     0.3
 -18133.1355    723.3731  4699  16:36:16     0.2
 -18131.3825    714.2407  4699  16:36:16     0.2
 -18130.8863    704.0639  4699  16:36:17     0.2
 -18130.8809    702.4212  4699  16:36:17     0.2
 -18130.8809    702.3588  4699  16:36:17     0.2
 -18130.8809    702.3570  4699  16:36:17     0.2
```

```
Finished on: Mon Jul 15 16:36:17 2013

LogLikelihood Converged
```

We note that `asreml()` uses the gamma parameterisation for fitting this model, as this is a single section, univariate analysis.

We fit the reduced animal model which requires inclusion of a term for the Mendelian sampling effects, and the fitting of male and female parents as outlined in equation 3.2.6. The portion of the script and the results are presented below.

```
> op1.df$zero <- rep(0,nrow(op1.df))
> op1.ram.sv <-
+     asreml(dbh~1,random=~ped(Fcln):zero + and(ped(Fcln),0.5) +
+            and(ped(Mcln),0.5) + Replicate + Replicate:Setgroup +
+            vv:units,
+            rcov=~units,data=op1.df,start.values=T,
+            ginverse=list(Fcln=op1.parents.ainv,Mcln=op1.parents.ainv),
+            na.method.X='include',workspace=80e6,sparse=~Check)
> op1.ram.sv$gammas.table

                            Gamma Value Constraint
1          ped(Fcln):zero!zero.var   0.1          P
2          Replicate!Replicate.var   0.1          P
3 Replicate:Setgroup!Replicate.var   0.1          P
4                  vv:units!vv.var   0.1          P
5                       R!variance   1.0          P

> gam <- op1.ram.sv$gammas.table[-5,]
> gam$fac <- factor(c(1,2,3,1))
> M <- asreml.constraints(~fac, gammas=gam)
> M

                                fac1 fac2 fac3
ped(Fcln):zero!zero.var            1    0    0
Replicate!Replicate.var            0    1    0
Replicate:Setgroup!Replicate.var   0    0    1
vv:units!vv.var                    1    0    0
attr(,"assign")
[1] 1 1 1
attr(,"contrasts")
attr(,"contrasts")$fac
[1] "contr.treatment"

> op1.ram.asr <-
+     asreml(dbh~1,random=~ped(Fcln):zero + and(ped(Fcln),0.5) +
+            and(ped(Mcln),0.5) + Replicate + Replicate:Setgroup +
+            vv:units,
```

```
+            rcov=~units,data=op1.df,
+            ginverse=list(Fcln=op1.parents.ainv,Mcln=op1.parents.ainv),
+            na.method.X='include',G.param=gam,constraints=M,
+            workspace=80e6,sparse=~Check)

asreml 3.0-1 (24 October 2012), Library: 3.0hj (15 November 2011), X86_64
     LogLik         S2      DF      wall     cpu
 -18135.5746    729.6107  4699  16:36:19     0.3
 -18133.1355    723.3731  4699  16:36:19     0.2
 -18131.3825    714.2407  4699  16:36:19     0.2
 -18130.8863    704.0639  4699  16:36:20     0.2
 -18130.8809    702.4212  4699  16:36:20     0.2
 -18130.8809    702.3588  4699  16:36:20     0.2
 -18130.8809    702.3570  4699  16:36:20     0.2

Finished on: Mon Jul 15 16:36:20 2013

LogLikelihood Converged
```

This set of commands and output requires careful explanation. The first term in the `random` model formula illustrates the use of the `and` function. The design matrix associated with $\boldsymbol{u}_{a_p}$, namely $\frac{1}{2}(\boldsymbol{F}_{op} + \boldsymbol{M}_{op})$ is formed using three adjacent model terms, the design matrices for the last two terms being added to the design matrix for the first term by use of `and()`. We note that due to a shortcoming of the current syntax, we have to create a *place-holder* so that we can overlay the female and male design matrices, with the 0.5 multipliers. This place-holder is conveniently created as a design matrix of all zeroes of the correct dimension (ie. number of effects consistent with following terms). This short-coming will be resolved in version 4 of **ASReml-R** with the generalisation of the `and()` function to allow for specification of a term consisting of a general linear function of model terms.

The other two features which are illustrated are the `start.values` argument and the use of the `asreml.constraints()` function. The `start.values` argument is a very useful facility which we use extensively for setting initial values as well as for applying constraints on the variance parameters. Setting this argument to `T` in the call to `asreml()` invokes the so-called template model of `asreml()` where the returned object has three components: `R.param, G.param` and `gammas.table`. The `gammas.table` component of this object is a data frame with three columns: `Gammas, Value`, and `Constraint` which are the name, initial value and constraint code, respectively, for each of the variance parameters in the linear mixed model. The initial values and constraints can be altered in `gammas.table`, and the data frame subsequently used with the `R.param` and `G.param` arguments. Note that setting `start.values` to a text string will also create a comma separated text file (of that name) in the form of `gammas.table`. This file can be edited and used with `R.param` and `G.param` in preference to `gammas.table` if desired.

The other feature which is used in the above analysis using the RA model is the constraints facilities within `asreml()`. Constraints on variance parameters, both within and between terms can be created and passed to `asreml()` through the use of the `asreml.constraints()` function

and the `R.param, G.param` and `constraints` arguments. In our example we form a 4×3 matrix $\boldsymbol{M}$ which contains the constraints which are placed on the variance parameters associated with the consolidated random model terms. We wish to constrain the variance parameters for the first and fourth model terms to be equal, and this is achieved by creating a factor, named `fac`, which has three levels: level 1 assigned to the first and fourth variance parameter, levels 2 and 3 to the second and third variance parameter.

Additional factors or variates created in `gammas.table`, and named in the `formula` argument to `asreml.constraints()`, can be used to create $\boldsymbol{M}$, defining the pattern of constraints applied to either as sigmas or gammas. The elements of $\boldsymbol{M}$ can be *real* numbers, but we note that each row of $\boldsymbol{M}$ can only have one non-zero element.

In our example, since we are only applying constraints to variance parameters which are associated with terms in the random model formula we need only specify the `G.param` argument. Both model terms `ped(Fcln)` and `ped(Mcln)` share the same inverse relationship matrix and this matrix is provided using the `ginverse` argument, being a list with named elements being the base factor name(s) used in the random model formula.

Lastly we fit the approximate reduced animal model. This is more straightforward to fit and is given by

```
> op1.aram.asr <-
+     asreml(dbh~1,random=~ped(Fcln):zero + and(ped(Fcln),0.5) +
+             and(ped(Mcln),0.5) + Replicate + Replicate:Setgroup,
+             rcov=~units,data=op1.df,
+             ginverse=list(Fcln=op1.parents.ainv,Mcln=op1.parents.ainv),
+             na.method.X='include',
+             workspace=80e6,sparse=~Check)

asreml 3.0-1 (24 October 2012), Library: 3.0hj (15 November 2011), X86_64
     LogLik         S2      DF      wall     cpu
 -18135.8387    764.3765  4699  16:36:21     0.1
 -18133.1268    764.6667  4699  16:36:21     0.0
 -18131.3022    764.8326  4699  16:36:21     0.0
 -18130.8535    764.3130  4699  16:36:21     0.0
 -18130.8500    763.9767  4699  16:36:21     0.0
 -18130.8500    763.9675  4699  16:36:21     0.0
 -18130.8500    763.9673  4699  16:36:21     0.0

Finished on: Mon Jul 15 16:36:21 2013

LogLikelihood Converged
```

The following presents a simple summary of the three models. The REML log-likelihoods for the full tree and the RA models are equivalent and the REML estimates of the variance parameters are the same. Note that the variance components for the additive genetic effects and the Mendelian sampling term are the same, as required. The REML estimates of the variance

parameters for the ARA model are very similar to the full tree and RA models with the REML estimate of the composite residual variance $\hat{\sigma}_a^{2*} = \hat{\sigma}_a^2/2 + \hat{\sigma}_e^{2*}$ where σ_e^{2*} is the pooled variance of the residual genetic effect and the residual errors.

```
> t1 <- summary(op1.tree.asr)$varcomp[,'component']
> t2 <- summary(op1.ram.asr)$varcomp[,'component']
> t3 <- summary(op1.aram.asr)$varcomp[,'component']
> op1.gams <- cbind(c(t1[1:3],NA,t1[4]),t2,c(t3[1:3],NA,t3[4]))
> dimnames(op1.gams) <- list(c('addvar','reps','rep:set','mendelian','residual'),
+                            c('Tree model','RA model','ARA model'))
> op1.gams

          Tree model  RA model ARA model
addvar     124.04028 124.04028 124.32924
reps        53.93712  53.93712  53.91746
rep:set     52.47757  52.47757  52.48191
mendelian         NA 124.04028        NA
residual   702.35702 702.35702 763.96728
```

4.2.4 Breeding values and reliabilities

Here we illustrate how to obtain (additive) breeding values (ie. E-BLUPs) for the additive genetic effects. The term *E-BLUP* refers to an Empirical Best Linear Unbiased Predictor, to denote the fact that we replace the unknown variance parameters by their REML estimates, distinguishing these from the BLUP which uses the true (but unknown) variance parameters. We use the `summary.asreml()` function to extract the E-BLUPs for all individuals (for the full tree model), the parental E-BLUPs for the RA model, and then combining these into a single data frame useful presentation and for comparison.

```
> op1.tree.cc <- summary(op1.tree.asr,all=T)$coef.random
> op1.aram.cc <- summary(op1.aram.asr,all=T)$coef.random
> op1.tree.cc <- as.data.frame(op1.tree.cc[grep('ped.*',
+                                               dimnames(op1.tree.cc)[[1]]),])
> op1.tree.cc$Tree <- attr(op1.ainv,'rowNames')
> names(op1.tree.cc) <- c('bv','se','zrat','Tree')
> op1.aram.cc <- as.data.frame(op1.aram.cc[grep('ped.*',
+                                               dimnames(op1.aram.cc)[[1]]),])
> op1.aram.cc$Tree <- attr(op1.parents.ainv,'rowNames')
> names(op1.aram.cc) <- c('bv','se','zrat','Tree')
> op1.both.bvs <- merge(op1.aram.cc,op1.tree.cc,by='Tree')
> names(op1.both.bvs)<- c('Tree','bv.aram','se.aram','zrat.aram',
+ 'bv.tree','se.tree','zrat.tree')
> op1.both.bvs <- subset(op1.both.bvs,!is.element(Tree,control.numbers))
> #xyplot(bv.aram~bv.tree,data=op1.both.bvs)
```

We then compute the reliabilities and augment the data frame with these.

```
> op1.both.bvs$rel.aram <- with(op1.both.bvs,1 -
+                               se.aram^2/op1.gams['addvar','ARA model'])
> op1.both.bvs$rel.tree <- with(op1.both.bvs,1 -
+                               se.tree^2/op1.gams['addvar','Tree model'])
```

4.3 Example 2: Analysis of a Clonal trial

4.3.1 Preliminaries: data construction

In this we present an analysis of a clonal trial using the models presented in chapter 3. The notes will closely follow the associated script file (`cloneop.R`) which has been distributed with the notes and data-sets on the USB stick. As for the first example presented in section 4.2, we have taken a standard approach in developing these scripts and included the pre-processing R commands which read in the data and set-up the basic data frame and structures which are necessary for the script file to run without dependence on other script files. For brevity we do not present or describe this header material again. We begin by creating three data frames, one for each trial. We will only consider the analysis of the first trial (ie. FR353_1, or `clone1.df`). As before, we use a supplied function to create pedigree data frames for all individuals and parental trees for each of the three trials.

```
> clone.sites <- c('FR353_1','FR353_2', 'FR353_3')
> clone1.df <- subset(ws.df,is.element(Expt,clone.sites[1]))
> clone2.df <- subset(ws.df,is.element(Expt,clone.sites[2]))
> clone3.df <- subset(ws.df,is.element(Expt,clone.sites[3]))
> temp.ped <- data.ped.trim(pedfile=wsped.df,data=clone1.df,parent=TRUE)
> clone1.tree.df <- temp.ped$tree.ped #619
> clone1.parent.df <- temp.ped$parent.ped #37
> temp.ped <- data.ped.trim(pedfile=wsped.df,data=clone2.df,parent=TRUE)
> clone2.tree.df <- temp.ped$tree.ped #625
> clone2.parent.df <- temp.ped$parent.ped #37
> temp.ped <- data.ped.trim(pedfile=wsped.df,data=clone3.df,parent=TRUE)
> clone3.tree.df <- temp.ped$tree.ped #655
> clone3.parent.df <- temp.ped$parent.ped #36
```

There are 619 trees in the full pedigree data frame for `clone1.df` compared to only 37 individuals in the parental data frame.

We also create the `Check` factor using similar script commands as for the OP trial. There are 48 levels in the factor `Check`, which is equal to one more than the number of trees which are progeny of control trees.

4.3.2 Forming pedigrees and related objects for the RA and ARA models

In this section we form the objects and additional columns within the data frame which are used in the subsequent analyses. Since most of this process is the same as presented for the analysis of the OP trial we provide only the script commands which are relevant to the analysis. The

portion of the script to produce the sparse form of the inverse relationship matrices for both the full and parental pedigrees is presented below.

```
> parents.clone1 <- clone1.parent.df$Tree
> nonparents.clone1 <- clone1.tree.df$Tree
> nonparents.clone1 <- nonparents.clone1[!is.element(nonparents.clone1,parents.clone1)]
> c(length(parents.clone1),length(nonparents.clone1))

[1]  37 582

> temp <- asreml.Ainverse(clone1.parent.df)
> table(temp$inbreeding)

 0
37

> clone1.parents.ainv <- temp$ginv
> temp <- asreml.Ainverse(clone1.tree.df)
> table(temp$inbreeding)

  0
619

> clone1.ainv <- temp$ginv
```

Again, there is little inbreeding for the individuals used in this trial (in fact none). The elements of $\boldsymbol{D}_{oo}$ are computed using similar commands as before but the details are not presented. The additional column `vv` contains the square root of the diagonal elements of $\boldsymbol{D}_{oo}$ with values given by $\sqrt{2}$ for all clonal trees (ie. not progeny of control parents).

4.3.3 Analyses using the Full tree, RA and ARA models

We now turn to the analyses using `asreml()`. The script file contains additional lines of code factorising all variables which may be used in the subsequent analyses as factors. We do not present this for brevity.

We now fit the full tree model, noting that the trial was designed as an incomplete block design, hence we include nested effects of `Iblk` within `Replicate`. The `ginverse` argument associates the `clone1.ainv` object with the `ped(Tree)` model term. Note also that we include `ide(Tree)`. This term creates a copy of the `Tree` factor, with the same number of levels as the `ped(Tree)` term but does not associate the additive relationship matrix listed in the `ginverse` argument. The variance matrix of this term is given by $\sigma^2_{\dot{e}}\boldsymbol{I}_m$ and therefore fits the residual genetic effects associated with the individuals. Note that since there is no data on parental trees the E-BLUPS for the parental trees is zero.

```
> clone1.tree.asr <- asreml(dbh~1,random=~ped(Tree) +
+                         Replicate + Replicate:Iblk + ide(Tree),
```

```
+                        rcov=~units,data=clone1.df,ginverse=list(Tree=clone1.ainv),
+                        na.method.X='include',
+                        workspace=80e6,sparse=~Check)

asreml 3.0-1 (24 October 2012), Library: 3.0hj (15 November 2011), X86_64
     LogLik         S2      DF      wall     cpu
 -10263.1857    896.9889  2579  16:36:23     0.1 (1 restrained)
 -10246.9298    864.5630  2579  16:36:23     0.0
 -10235.1676    826.8128  2579  16:36:23     0.0
 -10230.5021    795.2199  2579  16:36:23     0.0
 -10230.3176    790.4565  2579  16:36:23     0.0
 -10230.3130    790.1950  2579  16:36:23     0.0
 -10230.3130    790.1827  2579  16:36:23     0.0
 -10230.3130    790.1820  2579  16:36:23     0.0

Finished on: Mon Jul 15 16:36:23 2013

LogLikelihood Converged
```

Again `asreml()` uses the gamma parameterisation for fitting this model.

We now fit the reduced animal model which requires inclusion of a term for the Mendelian sampling effects, and the fitting of male and female parents. The key difference is that we fit `vv:Tree` rather than `vv:units` since the Mendelian sampling term must be fitted at the clone level, not at the residual, given we now have *true* replication. The portion of the script and the results are presented below.

```
> clone1.df$zero <- rep(0,nrow(clone1.df))
> clone1.ram.sv <-
+     asreml(dbh~1,random=~ped(Fcln):zero + and(ped(Fcln),0.5) +
+            and(ped(Mcln),0.5) + Replicate + Replicate:Iblk + Tree +
+            vv:Tree,
+            rcov=~units,data=clone1.df,start.values=T,
+            ginverse=list(Fcln=clone1.parents.ainv,Mcln=clone1.parents.ainv),
+            na.method.X='include',workspace=80e6,sparse=~Check)
> gam <- clone1.ram.sv$gammas.table[-6,]
> gam$fac <- factor(c(1,2,3,4,1))
> M <- asreml.constraints(~fac, gammas=gam)
> clone1.ram.asr <-
+     asreml(dbh~1,random=~ped(Fcln):zero + and(ped(Fcln),0.5) +
+            and(ped(Mcln),0.5) + Replicate + Replicate:Iblk + Tree +
+            vv:Tree,
+            rcov=~units,data=clone1.df,
+            ginverse=list(Fcln=clone1.parents.ainv,Mcln=clone1.parents.ainv),
+            na.method.X='include',G.param=gam,constraints=M,
+            workspace=80e6,sparse=~Check)

asreml 3.0-1 (24 October 2012), Library: 3.0hj (15 November 2011), X86_64
     LogLik         S2      DF      wall     cpu
```

```
 -10263.1857    896.9889  2579  16:36:24     0.0 (1 restrained)
 -10246.9298    864.5630  2579  16:36:24     0.0
 -10235.1676    826.8128  2579  16:36:24     0.0
 -10230.5021    795.2199  2579  16:36:24     0.0
 -10230.3176    790.4565  2579  16:36:24     0.0
 -10230.3130    790.1950  2579  16:36:24     0.0
 -10230.3130    790.1827  2579  16:36:24     0.0
 -10230.3130    790.1820  2579  16:36:24     0.0

Finished on: Mon Jul 15 16:36:24 2013

LogLikelihood Converged
```

Lastly we fit the approximate reduced animal model. This is more straightforward to fit and is given by

```
> clone1.aram.asr <-
+     asreml(dbh~1,random=~ped(Fcln):zero + and(ped(Fcln),0.5) +
+             and(ped(Mcln),0.5) + Replicate + Replicate:Iblk + Tree,
+             rcov=~units,data=clone1.df,
+             ginverse=list(Fcln=clone1.parents.ainv,Mcln=clone1.parents.ainv),
+             na.method.X='include',
+             workspace=80e6,sparse=~Check)

asreml 3.0-1 (24 October 2012), Library: 3.0hj (15 November 2011), X86_64
     LogLik          S2      DF      wall     cpu
 -10283.0040    938.7153  2579  16:36:25     0.0 (1 restrained)
 -10257.9172    892.7339  2579  16:36:25     0.0
 -10238.6236    841.1608  2579  16:36:25     0.0
 -10230.6283    798.4161  2579  16:36:25     0.0
 -10230.3173    790.6393  2579  16:36:25     0.0
 -10230.3130    790.1967  2579  16:36:25     0.0
 -10230.3130    790.1827  2579  16:36:25     0.0
 -10230.3130    790.1820  2579  16:36:25     0.0

Finished on: Mon Jul 15 16:36:25 2013

LogLikelihood Converged
```

The following presents a simple summary of the three models. The REML log-likelihoods for the full tree and the RA models are equivalent and REML estimates of the variance parameters are the same. Note that the variance components for the additive genetic effects and the Mendelian sampling term are the same, as required. The REML estimates of the variance parameters for the ARA model are almost exactly the same as the full tree and RA models which we expect given zero inbreeding in the clonal individuals. The REML estimate of the composite residual genetic variance is $\hat{\sigma}_a^{2*} = \hat{\sigma}_a^2/2 + \hat{\sigma}_e^2$.

```
> t1 <- summary(clone1.tree.asr)$varcomp[,'component']
```

```
> t2 <- summary(clone1.ram.asr)$varcomp[,'component']
> t3 <- summary(clone1.aram.asr)$varcomp[,'component']
> clone1.gams <- cbind(c(t1[1:4],NA,t1[5]),t2,c(t3[1:4],NA,t3[5]))
> dimnames(clone1.gams) <- list(c('addvar','reps','rep:ib','nonaddvar',
+                                 'mendelian','residual'),
+                               c('Tree model','RAM','ARA model'))
> clone1.gams

          Tree model        RAM  ARA model
addvar    256.269906 256.269906 256.269937
reps        3.927678   3.927678   3.927678
rep:ib     81.639544  81.639544  81.639545
nonaddvar 166.868374 166.868374 295.003324
mendelian         NA 256.269906         NA
residual  790.181994 790.181994 790.181990
```

4.3.4 Breeding values and reliabilities

Here we illustrate how to obtain (additive) breeding values (ie. E-BLUPs) for the additive genetic effects. We use the `summary.asreml()` function to extract the E-BLUPs for all individuals (for the full tree model), the parental E-BLUPs for the ARA model, and then combining these into a single data frame useful presentation and for comparison. We could also obtain the residual genetic effects for the clonal trees and add these to the additive effects to obtain E-BLUPs of the total genetic effects. The code also presents the computation of the reliabilities.

```
> clone1.tree.cc <- summary(clone1.tree.asr,all=T)$coef.random
> clone1.aram.cc <- summary(clone1.aram.asr,all=T)$coef.random
> clone1.tree.cc <- as.data.frame(clone1.tree.cc[grep('ped.*', dimnames(clone1.tree.cc)[[1]]),])
> clone1.tree.cc$Tree <- attr(clone1.ainv,'rowNames')
> names(clone1.tree.cc) <- c('bv','se','zrat','Tree')
> clone1.aram.cc <- as.data.frame(clone1.aram.cc[grep('ped.*',dimnames(clone1.aram.cc)[[1]]),])
> clone1.aram.cc$Tree <- attr(clone1.parents.ainv,'rowNames')
> names(clone1.aram.cc) <- c('bv','se','zrat','Tree')
> clone1.both.bvs <- merge(clone1.aram.cc,clone1.tree.cc,by='Tree')
> names(clone1.both.bvs)<- c('Tree','bv.aram','se.aram','zrat.aram','bv.tree','se.tree','zrat.tree')
> clone1.both.bvs <- subset(clone1.both.bvs,!is.element(Tree,control.numbers))
> #xyplot(bv.aram~bv.tree,data=clone1.both.bvs)
> clone1.both.bvs$rel.aram <- with(clone1.both.bvs,1 -
+                                  se.aram^2/clone1.gams['addvar','ARA model'])
> clone1.both.bvs$rel.tree <- with(clone1.both.bvs,1 -
+                                  se.tree^2/clone1.gams['addvar','Tree model'])
```

4.3.5 Forward selections: Prediction of breeding values for clones

It is often of interest when analysing clonal trials to produce predicted breeding values for the clones themselves. These are commonly known as forward selection breeding values. One of the potential drawbacks of an analysis based on the ARA model is that we do not obtain the full breeding value for non-parental individuals, such as clones. The Mendelian sampling effect for

these trees is confounded with the residual genetic effect. In this section we present an approach which overcomes this problem. Essentially, we simply augment the parental pedigree file with the trees for which we require forward selections and change the columns in the data frame which contain the male and female parent identifiers to be the tree itself for those trees we are interested in. In this example, and for most clonal trials, this is (nearly) equivalent to fitting the full tree model and so seems pointless. This, however is not the case when we are considering the analysis of multi-environment data-sets which consist of a mixture of OP, CP and clonal trials, as is the case for the analysis of the full RPBC data-set. In such an example, use of the ARA model, modified to allow for forward selection of *ALL* clonal trees, affords substantial savings in computational burden. At this stage such an analysis using a full tree model is not feasible.

Below are the script commands which are required to augment the parental pedigree data frame with the clones which require forward selections (in this case all the clones). As a precaution we also create a copy of the original columns of the data frame which contain the true (original coding of) male and female parents.

```
> clone1.df$Fcln.orig <- clone1.df$Fcln
> clone1.df$Mcln.orig <- clone1.df$Mcln
> temp.df <- subset(clone1.df,Check==1)
> length(unique(temp.df$Tree)) # 535

[1] 535

> temp.df <- subset(temp.df,!duplicated(Tree))[,names(clone1.parent.df)]
> head(temp.df)

     Tree   Fcln   Mcln
1  399001 880740 268041
6  399002 880740 268041
11 399004 880740 268041
16 399005 880740 268041
20 399006 880740 268041
21 399007 880740 268041

> temp.ped <- rbind(clone1.parent.df,temp.df[,names(clone1.parent.df)])
> nrow(temp.ped) # 572 = 37 + 535

[1] 572

> c(nrow(temp.ped),nrow(temp.ped[!duplicated(temp.ped$Tree),])) # [1] 572 572

[1] 572 572

> clone1.fsparent.df <- temp.ped
```

The next step involves changing the `Fcln` and `Mcln` columns and computing the sparse form of the inverse of the additive relationship matrix as such:

```
> temp <- as.character(clone1.df$Fcln)
> temp[clone1.df$Check==1] <- as.character(clone1.df$Tree)[clone1.df$Check==1]
> clone1.df$Fcln <- factor(temp)
> temp <- as.character(clone1.df$Mcln)
> temp[clone1.df$Check==1] <- as.character(clone1.df$Tree)[clone1.df$Check==1]
> clone1.df$Mcln <- temp
> temp <- asreml.Ainverse(clone1.fsparent.df)
> table(temp$inbreeding)

  0
572

> clone1.fsparents.ainv <- temp$ginv
```

The call to `asreml()` is much the same as for the previous analysis using the ARA model and is given here:

```
> clone1.fsaram.asr <-
+     asreml(dbh~1,random=~ped(Fcln):zero + and(ped(Fcln),0.5) +
+            and(ped(Mcln),0.5) + Replicate + Replicate:Iblk + Tree,
+            rcov=~units,data=clone1.df,
+            ginverse=list(Fcln=clone1.fsparents.ainv,Mcln=clone1.fsparents.ainv),
+            na.method.X='include',
+            workspace=80e6,sparse=~Check)

asreml 3.0-1 (24 October 2012), Library: 3.0hj (15 November 2011), X86_64
     LogLik         S2      DF      wall     cpu
 -10263.1857    896.9889  2579  16:36:26     0.0 (1 restrained)
 -10246.9298    864.5630  2579  16:36:26     0.0
 -10235.1676    826.8128  2579  16:36:26     0.0
 -10230.5021    795.2199  2579  16:36:26     0.0
 -10230.3176    790.4565  2579  16:36:26     0.0
 -10230.3130    790.1950  2579  16:36:26     0.0
 -10230.3130    790.1827  2579  16:36:26     0.0
 -10230.3130    790.1820  2579  16:36:26     0.0

Finished on: Mon Jul 15 16:36:26 2013

LogLikelihood Converged
```

To complete this section we simply provide some commands which compare the E-BLUPs for the first clonal tree in the data frame.

```
> cc <- coef(clone1.tree.asr,list=T)
> ccped <- cc[[grep('ped',names(cc))]]
> ccide <- cc[[grep('ide',names(cc))]]
> ccped[grep('.*399001',dimnames(ccped)[[1]]),] # me, [1] 14.02648

[1] 14.02648
```

```
> ccped[grep('.*880740',dimnames(ccped)[[1]]),] # mother, [1] 2.844393

[1] 2.844393

> ccped[grep('.*268041',dimnames(ccped)[[1]]),] # father, [1] 15.74325

[1] 15.74325

> ccide[grep('.*399001',dimnames(ccide)[[1]]),] # ide, [1] 6.163481

[1] 6.163481

> cc <- coef(clone1.ram.asr,list=T)
> ccped <- cc[[grep('ped',names(cc))]]
> ccide <- cc[[5]]
> ccmen <- cc[[grep('vv',names(cc))]]
> ccmen[grep('.*399001',dimnames(ccmen)[[1]]),] # mendelian, [1] 6.69299

[1] 6.69299

> ccped[grep('.*880740',dimnames(ccped)[[1]]),] # mother, [1] 2.844415

[1] 2.844393

> ccped[grep('.*268041',dimnames(ccped)[[1]]),] # father, [1] 15.74332

[1] 15.74325

> ccide[grep('.*399001',dimnames(ccide)[[1]]),] # ide, [1] 6.163481

[1] 6.163481

> (2.844415 + 15.74332)/2 + 6.693094/sqrt(2) # [1] 14.0266

[1] 14.0266

> cc <- coef(clone1.aram.asr,list=T)
> ccped <- cc[[grep('ped',names(cc))]]
> ccide <-cc[[grep('Tree',names(cc))]]
> ccped[grep('.*880740',dimnames(ccped)[[1]]),] # mother, [1] 2.844395

[1] 2.844395

> ccped[grep('.*268041',dimnames(ccped)[[1]]),] # father, [1] 15.74326

[1] 15.74326

> ccide[grep('.*399001',dimnames(ccide)[[1]]),] # ide, [1] 10.89614
```

```
[1] 10.89614

> (2.844395 + 15.74326)/2 # [1] 9.293827

[1] 9.293827

> cc <- coef(clone1.fsaram.asr,list=T)
> ccped <- cc[[grep('ped',names(cc))]]
> ccide <- cc[[grep('Tree',names(cc))]]
> ccped[grep('.*399001',dimnames(ccped)[[1]]),] # me, [1] 14.02648

[1] 14.02648

> ccped[grep('.*880740',dimnames(ccped)[[1]]),] # mother, [1] 2.844393

[1] 2.844393

> ccped[grep('.*268041',dimnames(ccped)[[1]]),] # father, [1] 15.74325

[1] 15.74325

> ccide[grep('.*399001',dimnames(ccide)[[1]]),] # ide, [1] 6.163481

[1] 6.163481
```

5 Analysis of multi-environment tree breeding experiments

5.1 Introduction

In this chapter we build on the models and approaches presented in chapter 3 and present an overview of an approach to the analysis of multi-environment trials (METs) in the context of tree breeding. We begin by considering the aim of a MET analysis. Aside from from gaining an understanding of genotype by environment interaction (G×E) or more often additive genotype by environment interaction (A×E), a key motivation is often to produce a set of additive genetic main effects. The former question, ie. understanding G×E (or A×E) is a relatively straightforward problem and is easily achieved by appropriate modelling of the variance structure of the G×E (or A×E) effects (either the interaction effects or the effects nested within environments), using for example, the approaches of Smith et al. (2001) who advocate the use of factor analytic (FA) variance models. The latter aim is a more challenging issue. In this chapter, we therefore seek to provide the necessary framework for achieving both these aims. The structure of the chapter is as follows. We will begin by presenting extensions of the full tree model and approximate reduced animal model for both OP/CP and clonal METs in sections 5.2 and 5.3. In section 5.4 we present a summary of variance models which we have found useful (or not useful) for the analysis of METs. In section 5.5 we consider a joint approach for METs with both OP/CP and clonal trials and conclude with a brief description of an approach we have developed to produce meaningful predictions which account for G×E. Key references for this chapter include Smith et al. (2001, 2005); Cullis et al. (2010); Beeck et al. (2010).

5.2 Statistical models for OP/CP METs

We now consider the analysis of a series of OP/CP trials, and let $\boldsymbol{y}^\intercal = (\boldsymbol{y}_1^\intercal, \boldsymbol{y}_2^\intercal, \ldots, \boldsymbol{y}_t^\intercal)$ be the combined data vector for t trials containing progeny from either OP or CP families. We refer to these trials as OP/CP trials and the full data-set as an OP/CP MET data-set. As before, we assume that the j^{th} trial comprises n_j plots (ie. observational units).

The statistical model for $\boldsymbol{y}$ can be written as

$$\boldsymbol{y} = \boldsymbol{X}\boldsymbol{\tau} + \boldsymbol{Z}_g\boldsymbol{u}_g + \boldsymbol{Z}_p\boldsymbol{u}_p + \boldsymbol{e} \tag{5.2.1}$$

where

- $\boldsymbol{\tau} = (\boldsymbol{\tau}_1^\intercal, \boldsymbol{\tau}_2^\intercal, \ldots, \boldsymbol{\tau}_t^\intercal)^\intercal$ is a vector of fixed effects with associated design matrix $\boldsymbol{X}$ (assumed to have full column rank);
- $\boldsymbol{u}_g = (\boldsymbol{u}_{g1}^\intercal, \boldsymbol{u}_{g2}^\intercal, \ldots, \boldsymbol{u}_{gt}^\intercal)^\intercal$ is the $mt \times 1$ vector of random genetic effects with associated design matrix $\boldsymbol{Z}_g$;
- $\boldsymbol{u}_p = (\boldsymbol{u}_{p1}^\intercal, \boldsymbol{u}_{p2}^\intercal, \ldots, \boldsymbol{u}_{pt}^\intercal)^\intercal$ is a vector of random non-genetic (or peripheral) effects with associated design matrix $\boldsymbol{Z}_p$ and
- $\boldsymbol{e} = (\boldsymbol{e}_1^\intercal, \boldsymbol{e}_2^\intercal, \ldots, \boldsymbol{e}_t^\intercal)^\intercal$ is the vector of residuals.

In the simplest case the vector $\boldsymbol{\tau}$ comprises an overall mean (intercept) for each trial but may include other effects as necessary. The vector $\boldsymbol{u}_p$, for example, may include effects for blocking factors associated with each trial design. The composition of this vector varies between trials, depending on the design (ie. complete or incomplete block design), the use of sets and multi-tree plots.

As for chapter 3 we assume that the $\boldsymbol{u}_g, \boldsymbol{u}_p$ and $\boldsymbol{e}$ vectors of random effects are mutually independent, and distributed as multivariate Gaussian, with zero means. The variance matrices of the vectors of non-genetic effects is given by $\boldsymbol{G}_p = \oplus_{k=1}^{b} \sigma_{p_k}^2 \boldsymbol{I}_{q_k}$ where $b = b'$ (see section 1.1.3) is the number of components in $\boldsymbol{u}_p$ and q_k is the number of effects in (length of) $\boldsymbol{u}_{p_k}$. The variance matrix of the vector of residual errors is assumed to given by $\boldsymbol{R} = \oplus_{j=1}^{t} \sigma_j^2 \boldsymbol{I}_{n_j}$.

As before we partition the total genetic by environment effects $\boldsymbol{u}_g$ into additive and residual genetic by environment effects given by

$$\boldsymbol{u}_g = \boldsymbol{u}_a + \boldsymbol{u}_e \tag{5.2.2}$$

and as before these vectors $\boldsymbol{u}_s, s = g, a, e$ are partitioned into two sub-vectors, the first representing those trees with progeny, and the latter representing those trees without progeny. This partition is denoted by $\boldsymbol{u}_s^\intercal = (\boldsymbol{u}_{sp}^\intercal,\ \boldsymbol{u}_{so}^\intercal), s = g, a, e$. There is a conformal partitioning of the columns of $\boldsymbol{Z}_g$ given by $\boldsymbol{Z}_g = [\boldsymbol{Z}_{g_p}\ \boldsymbol{Z}_{g_o}]$. We note that for this partition to apply, $\boldsymbol{u}_s$ must be ordered environments within individuals. This ordering was adopted by Beeck et al. (2010), but the reverse of the order considered in Smith et al. (2001).

We assume that the variance matrices of the vectors of additive and non-additive genetic effects are given by

$$\begin{aligned} \text{var}\,(\boldsymbol{u}_a) &= \boldsymbol{A} \otimes \boldsymbol{G}_a \quad \text{and} \\ \text{var}\,(\boldsymbol{u}_e) &= \boldsymbol{I}_m \otimes \boldsymbol{G}_e \end{aligned}$$

where $\boldsymbol{A}$ is the numerator relationship matrix with partitioning conformal with $\boldsymbol{u}_a$. The matrices $\boldsymbol{G}_a$ and $\boldsymbol{G}_e$ are $t \times t$ symmetric positive (semi)-definite matrices and are generally referred to as the additive and residual genetic between environment variance matrices.

Substituting equation 5.2.2 into equation 5.2.1 gives

$$\boldsymbol{y} = \boldsymbol{X}\boldsymbol{\tau} + \boldsymbol{Z}_g\boldsymbol{u}_a + \boldsymbol{Z}_g\boldsymbol{u}_e + \boldsymbol{Z}_p\boldsymbol{u}_p + \boldsymbol{e} \tag{5.2.3}$$

Since there is no true replication in OP/CP METs and individuals within a trial occur once and only once in the full MET data-set equation 5.2.3 becomes

$$\boldsymbol{y} = \boldsymbol{X}\boldsymbol{\tau} + \boldsymbol{Z}_g\boldsymbol{u}_a + \boldsymbol{Z}_p\boldsymbol{u}_p + \boldsymbol{e}_e^* \tag{5.2.4}$$

where $\boldsymbol{e}_e^*$ is a composite residual vector, whose elements are the sum of the residual genetic by environment effects and residual errors. Furthermore, since

$$\boldsymbol{Z}_g(\boldsymbol{I}_m \otimes \boldsymbol{G}_e)\boldsymbol{Z}_g^\intercal = \oplus_{j=1}^t \sigma_{e_j}^2 \boldsymbol{I}_{n_j}$$

where $\sigma_{e_j}^2, j = 1, \ldots, t$ are the diagonal elements of $\boldsymbol{G}_e$ then

$$\begin{aligned} \text{var}\,(\boldsymbol{e}_e^*) &= \oplus_{j=1}^t (\sigma_{e_j}^2 + \sigma_j^2)\boldsymbol{I}_{n_j} \\ &= \oplus_{j=1}^t \sigma_{e_j}^{2*} \boldsymbol{I}_{n_j} \\ &= \boldsymbol{R}_e^* \end{aligned}$$

Hence the distribution of $\boldsymbol{y}$ for OP/CP METs is multivariate Gaussian with

$$\begin{aligned} \text{E}\,(\boldsymbol{y}) &= \boldsymbol{X}\boldsymbol{\tau} \\ \text{var}\,(\boldsymbol{y}) &= \sigma_a^2 \boldsymbol{Z}_g(\boldsymbol{A} \otimes \boldsymbol{G}_a)\boldsymbol{Z}_g^\intercal + \boldsymbol{Z}_p\boldsymbol{G}_p\boldsymbol{Z}_p^\intercal + \boldsymbol{R}_e^* \end{aligned} \tag{5.2.5}$$

5.2.1 Reduced Animal model for the analysis of OP/CP METs

In this section we develop the RA and ARA models for the analysis of OP/CP MET data-sets. In chapter 3 the final form of the RA model was derived by assuming that the data vector $\boldsymbol{y} = (\boldsymbol{y}_p^\intercal, \boldsymbol{y}_o^\intercal)^\intercal$. This is a reordering of the data vector in equation 5.2.1 and subsequent equations. This reordering simply changes the form of $\boldsymbol{R}$ and $\boldsymbol{R}_e^*$. However, to simplify the following we assume that there is no data on parental trees and hence $\boldsymbol{y} = \boldsymbol{y}_o$. Using a similar approach presented in section 3.2.1 leads to the following model for $\boldsymbol{y}$

$$\boldsymbol{y} = \boldsymbol{X}\boldsymbol{\tau} + (\boldsymbol{Z}_{g_p} + \boldsymbol{Z}_{g_o}\boldsymbol{T}_{op})\boldsymbol{u}_{a_p} + \boldsymbol{Z}_{g_o}\boldsymbol{u}_{m_o} + \boldsymbol{Z}_g\boldsymbol{u}_e + \boldsymbol{Z}_p\boldsymbol{u}_p + \boldsymbol{e} \tag{5.2.6}$$

where $\boldsymbol{Z}_p$ has been partitioned in a similar manner to before, according to parental and non-parental individuals and the variance matrix of the vector of Mendelian A×E effects is given by

$$\text{var}\,(\boldsymbol{u}_{m_o}) = \boldsymbol{D}_{oo} \otimes \boldsymbol{G}_a$$

Using a similar argument to before leads to the final RA model for $\boldsymbol{y} = \boldsymbol{y}_o$ as

$$\begin{aligned} \boldsymbol{y} &= \boldsymbol{X}\boldsymbol{\tau} + \frac{1}{2}(\boldsymbol{F}_{op} + \boldsymbol{M}_{op})\boldsymbol{u}_{a_p} + \boldsymbol{u}_{m_o} + \boldsymbol{u}_{e_o} + \boldsymbol{Z}_p\boldsymbol{u}_p + \boldsymbol{e} \\ &= \boldsymbol{X}\boldsymbol{\tau} + \frac{1}{2}(\boldsymbol{F}_{op} + \boldsymbol{M}_{op})\boldsymbol{u}_{a_p} + \boldsymbol{Z}_p\boldsymbol{u}_p + \boldsymbol{e}_a^* \end{aligned} \tag{5.2.7}$$

where $\boldsymbol{e}_a^*$ is a composite residual, whose elements are the sum of Mendelian and residual genetic by environment effects and residual errors. The variance of this composite residual is given by

$$\oplus_{j=1}^{t}(\sigma_{a_j}^2 \boldsymbol{D}_{oo_j} + \sigma_{e_j}^2 \boldsymbol{I}_{n_j} + \sigma_j^2 \boldsymbol{I}_{n_j})$$

for some matrix $\boldsymbol{D}_{oo_j}$ which is the submatrix of $\boldsymbol{D}_{oo}$ relating to those individuals in trial j. Ignoring inbreeding then the approximate variance matrix is

$$\oplus_{j=1}^{t}(\frac{1}{2}\sigma_{a_j}^2 + \sigma_{e_j}^2 + \sigma_j^2)\boldsymbol{I}_{n_j} = \oplus_{j=1}^{t}\sigma_{a_j}^{2*}\boldsymbol{I}_{n_j}$$

say. This is the ARA model for OP/CP MET data-sets.

5.3 Statistical models for clonal METs

We now consider the analysis of a series of clonal trials, and as before let $\boldsymbol{y}^\intercal = (\boldsymbol{y}_1^\intercal, \boldsymbol{y}_2^t, \ldots, \boldsymbol{y}_t^\intercal)$ be the combined data vector for t trials containing clonal progeny from CP families. We refer to these trials as clonal trials and the full data-set as a clonal MET data-set. As before, we assume that the j^{th} trial comprises n_j plots (ie. observational units).

The statistical model for $\boldsymbol{y}$ can be written as

$$\boldsymbol{y} = \boldsymbol{X}\boldsymbol{\tau} + \boldsymbol{Z}_g\boldsymbol{u}_g + \boldsymbol{Z}_p\boldsymbol{u}_p + \boldsymbol{e} \tag{5.3.8}$$

where all model terms have the same definition as for equation 5.2.1.

Substituting equation 5.2.2 into equation 5.3.8 gives

$$\boldsymbol{y} = \boldsymbol{X}\boldsymbol{\tau} + \boldsymbol{Z}_g\boldsymbol{u}_a + \boldsymbol{Z}_g\boldsymbol{u}_e + \boldsymbol{Z}_p\boldsymbol{u}_p + \boldsymbol{e} \tag{5.3.9}$$

Here there is true replication and individuals usually occur more than once within a trial and often occur in more than one trial. The distribution of $\boldsymbol{y}$ for clonal METs is multivariate Gaussian with

$$\begin{aligned} \mathrm{E}(\boldsymbol{y}) &= \boldsymbol{X}\boldsymbol{\tau} \\ \mathrm{var}(\boldsymbol{y}) &= \boldsymbol{Z}_g(\boldsymbol{A}\otimes\boldsymbol{G}_a)\boldsymbol{Z}_g^\intercal + \boldsymbol{Z}_g(\boldsymbol{I}_m\otimes\boldsymbol{G}_e)\boldsymbol{Z}_g^\intercal + \boldsymbol{Z}_p\boldsymbol{G}_p\boldsymbol{Z}_p^\intercal + \boldsymbol{R} \end{aligned} \tag{5.3.10}$$

5.3.1 Reduced Animal (RA) model for the analysis of clonal METs

In this section we develop the RA and ARA models for the analysis of clonal MET data-sets. Using a similar approach presented in section 5.2 leads to the following model for $\boldsymbol{y} = \boldsymbol{y}_o$

$$\begin{aligned} \boldsymbol{y} &= \boldsymbol{X}\boldsymbol{\tau} + \boldsymbol{Z}_{g_{oo}}\boldsymbol{T}_{op}\boldsymbol{u}_{a_p} + \boldsymbol{Z}_{g_{oo}}\boldsymbol{u}_{m_o} + \boldsymbol{Z}_{g_{oo}}\boldsymbol{u}_{e_o} + \boldsymbol{Z}_p\boldsymbol{u}_p + \boldsymbol{e} \\ &= \boldsymbol{X}\boldsymbol{\tau} + \frac{1}{2}(\boldsymbol{Z}_{f_{op}} + \boldsymbol{Z}_{m_{op}})\boldsymbol{u}_{a_p} + \boldsymbol{Z}_{g_{oo}}\boldsymbol{u}_{a_o}^* + \boldsymbol{Z}_p\boldsymbol{u}_p + \boldsymbol{e} \end{aligned} \tag{5.3.11}$$

where $\boldsymbol{u}_{a_o}^*$ is a composite genetic by environment residual, with elements being the sum of Mendelian and residual genetic by environment effects for non-parental individuals. The variance matrix of this composite genetic residual is given by

$$\boldsymbol{D}_{oo}\otimes\boldsymbol{G}_a + \boldsymbol{I}_{m_o}\otimes\boldsymbol{G}_e$$

Again, ignoring inbreeding, the approximate variance matrix is

$$\boldsymbol{I}_{m_o} \otimes (\frac{1}{2}\boldsymbol{G}_a + \boldsymbol{G}_e)$$

Given this, it may not be sensible to use the ARA model for the analysis of clonal MET data-sets. We have often found that, in crop breeding METs, the variance matrices for each component of the composite genetic residual to be quite different.

In practice, however, we often include clonal trees in the analysis as parents, ie. as forward selections and this avoids this problem. This may not be an option for larger clonal MET data-sets in which it is computationally prohibitive to fit the full tree or ARA models with forward selections.

5.4 Variance models useful for modelling G×E

In this section we present a brief overview of some variance models which we either find useful, or have been frequently used for modelling the variance structure of the G×E or A×E effects in MET data-sets. In the following we consider, as an example, modelling the variance structure of $\boldsymbol{u}_a$, the vector of additive by environment effects for the full tree model. The models we develop also directly apply to modelling of the residual genetic and parental additive by environment effects. Recall that since

$$\text{var}\,(\boldsymbol{u}_a) = \boldsymbol{A} \otimes \boldsymbol{G}_a$$

then we need only consider models for $\boldsymbol{G}_a$, since $\boldsymbol{A}$ is known. For $t = 2$ trials,

$$\boldsymbol{G}_a = \begin{bmatrix} \sigma^2_{a_1} & \sigma_{a_{12}} \\ \sigma_{a_{12}} & \sigma^2_{a_2} \end{bmatrix}$$

and for t trials,

$$\boldsymbol{G}_a = \begin{bmatrix} \sigma^2_{a_1} & \sigma_{a_{12}} & \sigma_{a_{13}} & \cdots & \sigma_{a_{1t}} \\ & \sigma^2_{a_2} & \sigma_{a_{23}} & \cdots & \sigma_{a_{2t}} \\ & & \sigma^2_{a_3} & \cdots & \sigma_{a_{3t}} \\ & & & \ddots & \vdots \\ & & & & \sigma^2_{a_t} \end{bmatrix}$$

This matrix is symmetric and must, in general, be positive (semi)-definite. The most general variance model is the so-called *unstructured* variance model. In `asreml()` the model is specified by the `us()` variance model function. This model has $p = t(t+1)/2$ parameters, since we assume symmetric $\boldsymbol{G}_a$. For example, if $t = 2$, then $p = 3$, while if $t = 60$ (more like the full RPBC MET data-set), then $p = 1830$. Clearly as t increases p becomes prohibitively large and this influences both our ability to fit the unstructured variance model, as well as the reliability of the estimates we obtain from a *successful* fit. Most often the REML estimate of the variance matrix may either be *singular* or we find that the average information algorithm used in **ASReml-R** fails to converge within the parameter space (ie. where the variance matrix is constrained, by default, to be positive definite).

A submodel of the unstructured variance model, which we often use in a preliminary analysis of MET data-sets is the so-called diagonal variance model, in which all of the covariances between

each pair of environments is assumed to be zero. This implies that $\boldsymbol{G}_a$ is a diagonal matrix, hence the name. `asreml()` (by default) constrains the diagonal elements of $\boldsymbol{G}_a$ to be positive. If one (or more) of the REML estimates of the variance components (ie. parameters of the `diag()` variance model) is fixed at zero, then it may then be difficult to fit other more complex variance models such as the unstructured or factor analytic variance models.

The unstructured variance model has been frequently used for the analysis of tree breeding MET data-sets. Baltunis et al. (2010); Apiolaza (2012); Raymond (2011) used this model in this context, but they mostly fitted the model to subsets of two trials from the full set of trials. Such an approach avoids the difficulty in fitting the model, but is statistically inefficient and can lead to an overall estimate of $\boldsymbol{G}_a$ which is not positive (semi)-definite.

Apart from the lack of parsimony of the unstructured variance model, the other difficulty with fitting this variance model to MET data-sets is the poor connectivity between trials. The example MET data-set described in chapter 2 has relatively good parental connectivity but many larger MET data-sets, particularly those found in tree breeding, often have poor parental connectivity. This has, in part, led to the approach adopted in the papers cited above, where the unstructured variance model is fitted to those pairs of trials having good connectivity.

These disadvantages with the unstructured variance model led to the use of other variance models. The simplest variance model used for modelling the variance structure of the G(A)×E interaction effects is the compound symmetric variance model. This variance model can be considered as the "traditional" variance components model, which involves fitting the main effects of both `Environment` (as a fixed effect) and `Tree`, as in this case `ped(Tree)` (as a random effect) and the interaction, in this case as `ped(Tree):Environment`. For this model, for $t = 2$ we have

$$\boldsymbol{G}_a = \begin{bmatrix} \sigma_a^2 + \sigma_{ae}^2 & \sigma_a^2 \\ \sigma_a^2 & \sigma_a^2 + \sigma_{ae}^2 \end{bmatrix}$$

and for t trials

$$\boldsymbol{G}_a = \begin{bmatrix} \sigma_a^2 + \sigma_{ae}^2 & \sigma_a^2 & \sigma_a^2 & \cdots & \sigma_a^2 \\ & \sigma_a^2 + \sigma_{ae}^2 & \sigma_a^2 & \cdots & \sigma_a^2 \\ & & \sigma_a^2 + \sigma_{ae}^2 & \cdots & \sigma_a^2 \\ & & & \ddots & \vdots \\ & & & & \sigma_a^2 + \sigma_{ae}^2 \end{bmatrix}$$

where σ_a^2 is the variance component for additive genetic main effects and σ_{ae}^2 is the variance component for A×E interactions. Thus the compound symmetric variance model implies that all trials have the same genetic variance and all pairs of trials have the same genetic covariance (and thence same genetic correlation). The advantages of this model is that it is easy to fit, can handle unbalanced data and incorporates a main effect. The disadvantage is that it rarely provides a good fit to the data.

We have found that the factor analytic (FA) variance model (Smith et al., 2001) provides a good approximation to the unstructured variance model (Kelly et al., 2007). The implementation in **ASReml-R**, due to Thompson et al. (2003) handles REML estimates of $\boldsymbol{G}_a$ which are less than

full rank and is computationally efficient. Perhaps the most useful derivation of the FA(k) (ie. an FA variance model of order k) variance model arises from a multiplicative/regression model for the genetic effects in each environment:

$$u_{a_{ij}} = \lambda_{a_{1j}} f_{a_{1i}} + \lambda_{a_{2j}} f_{a_{2i}} + \ldots \lambda_{a_{kj}} f_{a_{ki}} + \delta_{a_{ij}} \tag{5.4.12}$$

which involves a sum of multiplicative terms (known as "factors"). Each component is the product of an additive genetic effect ("score", $f_{a_{ri}}$) and an environment effect ("loading", $\lambda_{a_{rj}}$). The "order" (k) of the FA model is the number of number of factors. The final term represents a lack of fit of the regression model. The FA(1) variance model is a generalisation of the compound symmetric variance model, in that the compound symmetric variance model is an FA(1) variance model in which the loadings are all constrained to be equal, and the variance for the lack of fit term is assumed constant.

The model (ie. equation 5.4.12) can be written in matrix/vector notation

$$\boldsymbol{u}_a = (\boldsymbol{I}_m \otimes \boldsymbol{\Lambda}_a)\, \boldsymbol{f}_a + \boldsymbol{\delta}_a \tag{5.4.13}$$

where

- $\boldsymbol{\Lambda}_a$ is the $t \times k$ matrix of environment loadings,
- $\boldsymbol{f}_a$ is the $mk \times 1$ vector of additive genetic scores
- $\boldsymbol{\delta}_a$ is the $mt \times 1$ vector of additive genetic "residuals"

We assume $\boldsymbol{f}_a$ and $\boldsymbol{\delta}_a$ are independent and are distributed as multivariate Gaussian, zero means and

$$\text{var}\,(\boldsymbol{f}_a) = \boldsymbol{A} \otimes \boldsymbol{I}_k \qquad \text{and} \qquad \text{var}\,(\boldsymbol{\delta}_a) = \boldsymbol{A} \otimes \boldsymbol{\Psi}_a$$

where $\boldsymbol{\Psi}_a$ is a $t \times t$ diagonal matrix with a variance ("specific variance") for each environment. These assumptions lead to

$$\text{var}\,(\boldsymbol{u}_a) = (\boldsymbol{\Lambda}_a \boldsymbol{\Lambda}_a^{\intercal} + \boldsymbol{\Psi}_a) \otimes \boldsymbol{A}$$

The form of $\boldsymbol{G}_a$ for the FA(k) variance model is:

$$\begin{bmatrix} \sum_{r=1}^k \lambda_{a_{r1}}^2 + \psi_{a_1} & \sum_{r=1}^k \lambda_{a_{r1}}\lambda_{a_{r2}} & \sum_{r=1}^k \lambda_{a_{r1}}\lambda_{a_{r3}} & \cdots & \sum_{r=1}^k \lambda_{a_{r1}}\lambda_{a_{rt}} \\ & \sum_{r=1}^k \lambda_{a_{r2}}^2 + \psi_{a_2} & \sum_{r=1}^k \lambda_{a_{r2}}\lambda_{a_{r3}} & \cdots & \sum_{r=1}^k \lambda_{a_{r2}}\lambda_{a_{rt}} \\ & & \sum_{r=1}^k \lambda_{a_{r3}}^2 + \psi_{a_3} & \cdots & \sum_{r=1}^k \lambda_{a_{r3}}\lambda_{a_{rt}} \\ & & & \ddots & \\ & & & & \sum_{r=1}^k \lambda_{a_{rt}}^2 + \psi_{a_t} \end{bmatrix}$$

The total number of variance parameters in the FA(k) variance model is $(k+1)t - k(k-1)/2$. As t increases the number of parameters in an FA(k) variance model is linear in t compared to an unstructured variance model in which the number of parameters is quadratic in t. As mentioned earlier Kelly et al. (2007) found that FA variance models provide an excellent approximation to an unstructured variance model and often produce an improved fit.

The syntax for fitting an FA variance model is straightforward. For example to fit an FA(1) for our example, the consolidated (random) model term is `ped(Tree):fa(Environment,1)`. For $k > 1$ `asreml()` automatically applies constraints to the loadings matrix $\boldsymbol{\Lambda}_a$ by fixing elements in the upper triangle to the initial value (typically zero).

5.5 Joint approach for the analysis of tree breeding MET data-sets

We now develop a model for the analysis of a MET data-set which contains OP, CP and clonal trials. The model is a simple extension of the models presented in the previous sections for OP/CP MET data-sets and clonal MET data-sets and only requires merging of the data vectors and re-definition of the design matrices, vectors of fixed and random effects and the residual error term. We consider only the full tree model and the ARA model in the following sections.

5.5.1 Full tree model for combined MET data-sets

Without loss of generality we let $\boldsymbol{y} = (\boldsymbol{y}_1^\intercal, \boldsymbol{y}_2^\intercal)^\intercal$ be the data vector for the combined MET data-set, where $\boldsymbol{y}_j, j = 1, 2$ are the data vectors for the OP/CP MET data-set and the clonal MET data-set respectively. It therefore follows that the model for $\boldsymbol{y}$ is given by

$$\begin{bmatrix} \boldsymbol{y}_1 \\ \boldsymbol{y}_2 \end{bmatrix} = \begin{bmatrix} \boldsymbol{X}_1\boldsymbol{\tau}_1 \\ \boldsymbol{X}_2\boldsymbol{\tau}_2 \end{bmatrix} + \begin{bmatrix} \boldsymbol{Z}_{g_1}\boldsymbol{u}_{a_1} \\ \boldsymbol{Z}_{g_2}\boldsymbol{u}_{a_2} \end{bmatrix} + \begin{bmatrix} \boldsymbol{0} \\ \boldsymbol{Z}_{g_2}\boldsymbol{u}_{e_2} \end{bmatrix} + \begin{bmatrix} \boldsymbol{Z}_{p_1}\boldsymbol{u}_{p_1} \\ \boldsymbol{Z}_{p_2}\boldsymbol{u}_{p_2} \end{bmatrix} + \begin{bmatrix} \boldsymbol{e}^*_{e_1} \\ \boldsymbol{e}_2 \end{bmatrix} \tag{5.5.14}$$

This model can be written more succinctly as

$$\boldsymbol{y} = \boldsymbol{X}\boldsymbol{\tau} + \boldsymbol{Z}_g\boldsymbol{u}_a + \begin{bmatrix} \boldsymbol{0} \\ \boldsymbol{Z}_{g_2} \end{bmatrix} \boldsymbol{u}_{e_2} + \boldsymbol{Z}_p\boldsymbol{u}_p + \begin{bmatrix} \boldsymbol{e}^*_{e_1} \\ \boldsymbol{e}_2 \end{bmatrix} \tag{5.5.15}$$

where $\boldsymbol{X} = \text{diag}\left(\boldsymbol{X}_j\right), \boldsymbol{Z}_p = \text{diag}\left(\boldsymbol{Z}_{p_j}\right), \boldsymbol{Z}_g = \text{diag}\left(\boldsymbol{Z}_{g_j}\right)$ and $\boldsymbol{\tau} = (\boldsymbol{\tau}_1^\intercal, \boldsymbol{\tau}_2^\intercal)^\intercal, \boldsymbol{u}_p = (\boldsymbol{u}_{p_1}^\intercal, \boldsymbol{u}_{p_2}^\intercal)^\intercal, \boldsymbol{u}_a = (\boldsymbol{u}_{a_1}^\intercal, \boldsymbol{u}_{a_2}^\intercal)^\intercal$

5.5.2 ARA model for combined MET data-sets

Using the same approach as in section 5.5.1 the model for $\boldsymbol{y}$, the data vector for the combined MET data-set is

$$\begin{bmatrix} \boldsymbol{y}_1 \\ \boldsymbol{y}_2 \end{bmatrix} = \begin{bmatrix} \boldsymbol{X}_1\boldsymbol{\tau}_1 \\ \boldsymbol{X}_2\boldsymbol{\tau}_2 \end{bmatrix} + \begin{bmatrix} \frac{1}{2}(\boldsymbol{F}_{op_1} + \boldsymbol{M}_{op_1})\boldsymbol{u}_{a_{p_1}} \\ \frac{1}{2}(\boldsymbol{Z}_{f_{op_2}} + \boldsymbol{Z}_{m_{op_2}})\boldsymbol{u}_{a_{p_2}} \end{bmatrix} + \begin{bmatrix} \boldsymbol{0} \\ \boldsymbol{Z}_{g_{oo_2}}\boldsymbol{u}^*_{a_{o_2}} \end{bmatrix} + \begin{bmatrix} \boldsymbol{Z}_{p_1}\boldsymbol{u}_{p_1} \\ \boldsymbol{Z}_{p_2}\boldsymbol{u}_{p_2} \end{bmatrix} + \begin{bmatrix} \boldsymbol{e}^*_{a_1} \\ \boldsymbol{e}_2 \end{bmatrix} \tag{5.5.16}$$

This model can be written more succinctly as

$$\boldsymbol{y} = \boldsymbol{X}\boldsymbol{\tau} + \boldsymbol{Z}_g\boldsymbol{u}_{a_p} + \begin{bmatrix} \boldsymbol{0} \\ \boldsymbol{Z}_{g_2} \end{bmatrix} \boldsymbol{u}^*_{a_{o_2}} + \boldsymbol{Z}_p\boldsymbol{u}_p + \begin{bmatrix} \boldsymbol{e}^*_{a_1} \\ \boldsymbol{e}_2 \end{bmatrix} \tag{5.5.17}$$

where $\boldsymbol{X} = \text{diag}\left(\boldsymbol{X}_j\right), \boldsymbol{Z} = \text{diag}\left(\boldsymbol{Z}_{p_j}\right), \boldsymbol{\tau} = (\boldsymbol{\tau}_1^\intercal, \boldsymbol{\tau}_2^\intercal)^\intercal, \boldsymbol{u}_p = (\boldsymbol{u}_{p_1}^\intercal, \boldsymbol{u}_{p_2}^\intercal)^\intercal$ and

$$\boldsymbol{Z}_g = \begin{bmatrix} \frac{1}{2}(\boldsymbol{F}_{op_1} + \boldsymbol{M}_{op_1}) \\ \frac{1}{2}(\boldsymbol{Z}_{f_{op_2}} + \boldsymbol{Z}_{m_{op_2}}) \end{bmatrix}$$

Fitting these models in `asreml()` will be illustrated in chapter 6.

6 Analysis of MET tree breeding experiments: Examples

6.1 Introduction

In this chapter we present the analysis of three sets of MET tree breeding data-sets, taken from the six trials described in chapter 2. These data-sets are the OP/CP set, the clonal set and the complete set respectively. We will illustrate fitting the full tree and the ARA model using a range of variance models presented in section 5.4.

6.2 Example 3: Analysis of an OP/CP MET data-set

In this section we present an analysis of the OP/CP MET data-set which comprises three trials, namely FR203_1, FR203_2 and FR203_3. We will only present the results for the full tree and ARA models. Again, the notes will closely follow the associated script file (`metop.R`) which has been distributed with the notes and data-sets on the USB stick. This script file contains the pre-processing commands to ensure that it can be used independently of other scripts. We do not present these commands here as they are only slightly modified versions of the pre-processing commands in used in `ssop.R`.

The first step, after the pre-processing is to set up the parental and individual pedigree data-frames, along with the factor `Check`. There are 11954 individuals in the tree pedigree file and 298 parents. There are 1021 levels in `Check`.

6.2.1 Example 3: Full tree model

We now present the results for the full tree model, where we focus on modelling the A×E variance structure, using the models presented in chapter 5, namely the diagonal, compound symmetric and unstructured models. With only $t = 3$ trials, fitting an unstructured variance model may be less difficult and we note that we can only fit an FA(1) variance model. Both models have $p = 6$ parameters, though the fit may be (slightly) different for each, due to the different parameterisations. Therefore, unless we encounter difficulties in fitting an unstructured matrix we will not use an FA(1) variance model for $t = 3$ in the following examples.

The syntax and output for fitting the diagonal variance model is:

```
> ###################################
> # now we fit the tree model
> # All three OP trials are Set = 2 ie fit setgroup
> # none are IB, all are Plottype = 1
> # results are very close - discrepancy from convergence criteria
> # ie. should get the same as single site!
> # note that residual error variances are VERY different while
> # additive variance is quite constant!
> ###############################
> opmet.tree.asr <- asreml(dbh~Expt,random=~diag(Expt):ped(Tree) +
+                          at(Expt):Replicate +
+                          at(Expt):Replicate:Setgroup,
+                          rcov=~at(Expt):units,data=opmet.df,
+                          ginverse=list(Tree=opmet.ainv),na.method.X='include',
+                          workspace=80e6,sparse=~Check)

asreml 3.0-1 (24 October 2012), Library: 3.0hj (15 November 2011), X86_64
     LogLik          S2      DF      wall     cpu
 -41466.8796      1.0000 10633  21:37:01     1.1
 -41214.5390      1.0000 10633  21:37:01     0.8
 -40997.2462      1.0000 10633  21:37:02     0.8
 -40909.0257      1.0000 10633  21:37:03     0.8
 -40906.7068      1.0000 10633  21:37:04     0.8
 -40906.7034      1.0000 10633  21:37:05     0.8
 -40906.7034      1.0000 10633  21:37:05     0.8
 -40906.7034      1.0000 10633  21:37:06     0.8

Finished on: Mon Jul 15 21:37:06 2013

LogLikelihood Converged

> ss <- summary(opmet.tree.asr)$varcomp
> cbind(ss[grep('.*FR203_1',dimnames(ss)[[1]]),'component'],op1.gams[-4,'Tree model'])

              [,1]      [,2]
addvar   124.04027 124.04028
reps      53.93712  53.93712
rep:set   52.47757  52.47757
residual 702.35696 702.35702
```

This model is equivalent to a separate analysis of each of the three trials. This is because we fit a `diag()` variance model to each of the additive genetic by environment effects, and the residual errors. We also note, that the `at()` model constructor function has a dual role which we believe can lead to confusion. In this example, the term `at(Expt):Replicate` is a short-hand way of fitting three random model terms, each with a default (ie. IID) variance model. That is the expanded form is given by

```
at(Expt,1):Replicate + at(Expt,2):Replicate + at(Expt,3):Replicate
```

which is equivalent to

```
id(at(Expt,1)):idv(Replicate) + id(at(Expt,2)):idv(Replicate) +
id(at(Expt,3)):idv(Replicate)
```

which is also equivalent to

```
idv(at(Expt,1)):id(Replicate) + idv(at(Expt,2)):id(Replicate) +
idv(at(Expt,3)):id(Replicate)
```

or similarly

```
idv(at(Expt)):Replicate
```

This use of `at()` is at first glance, very much the same as fitting `diag(Expt):Replicate`. The most important difference, however, between these forms is that `asreml()` creates a new copy of `Replicate` for each level of `Expt` discarding redundancies (ie. levels which do not appear in the data.

A simple check of the fit of the diagonal variance model, can be done by comparing the REML estimates of this model with the single site analysis (of FR203_1) presented in chapter 4. This is done in the final two lines of code. The estimates agree within rounding errors.

We now consider fitting the compound symmetric variance model, which we fit by fitting default variance models to the main effect and interaction effects, namely `ped(Tree)` and `Environment:ped(Tree)` respectively.

```
> ######################
> # now other variance models for the
> # ped():Expt term.
> # only try G+G:E and us()
> # CS model
> ######################
> opmet.tree.asr1 <- asreml(dbh~Expt,random=~ped(Tree) + Expt:ped(Tree) +
+                           at(Expt):Replicate +
+                           at(Expt):Replicate:Setgroup,
+                           rcov=~at(Expt):units,data=opmet.df,
+                           ginverse=list(Tree=opmet.ainv),na.method.X='include',
+                           workspace=80e6,sparse=~Check)

asreml 3.0-1 (24 October 2012), Library: 3.0hj (15 November 2011), X86_64
     LogLik          S2      DF      wall     cpu
 -41255.6751      1.0000 10633  21:37:11     3.8
 -41089.5509      1.0000 10633  21:37:14     2.9
 -40950.4958      1.0000 10633  21:37:17     2.8
```

```
 -40897.1727      1.0000 10633  21:37:20     2.7
 -40896.0230      1.0000 10633  21:37:22     2.7
 -40896.0210      1.0000 10633  21:37:25     2.9
 -40896.0210      1.0000 10633  21:37:28     2.8
 -40896.0210      1.0000 10633  21:37:31     2.8

Finished on: Mon Jul 15 21:37:31 2013

LogLikelihood Converged
```

```
> ss1 <- summary(opmet.tree.asr1)$varcomp[,c('gamma','component')]
> ss1
```

```
                                                    gamma  component
ped(Tree)!ped                                    66.27401   66.27401
Expt:ped(Tree)!Expt.var                          39.49576   39.49576
at(Expt, FR203_1):Replicate!Replicate.var        55.57539   55.57539
at(Expt, FR203_2):Replicate!Replicate.var        34.55203   34.55203
at(Expt, FR203_3):Replicate!Replicate.var        66.82428   66.82428
at(Expt, FR203_1):Replicate:Setgroup!Replicate.var 52.27163   52.27163
at(Expt, FR203_2):Replicate:Setgroup!Replicate.var 52.56704   52.56704
at(Expt, FR203_3):Replicate:Setgroup!Replicate.var 25.33262   25.33262
Expt_FR203_1!variance                           713.14258  713.14258
Expt_FR203_2!variance                           472.43233  472.43233
Expt_FR203_3!variance                          1519.07372 1519.07372
```

Note that the two columns of the data frame produced by the `summary.asreml()` function are identical. This is because `asreml()` detects that this overall variance model is a multi-section variance model and hence defaults to the sigma parameterisation for fitting. This lingering anomaly in `summary.asreml()` will be resolved in the next version of `asreml()`.

The fit of the unstructured variance model is:

```
> ######################
> # us() model
> # aginst my better judgement I will use the default
> # starting values for us()
> # two odd things
> # converged and CS and us() are much the same!
> ######################
> opmet.tree.asr2 <- asreml(dbh~Expt,random=~us(Expt):ped(Tree) +
+                           at(Expt):Replicate +
+                           at(Expt):Replicate:Setgroup,
+                           rcov=~at(Expt):units,data=opmet.df,
+                           ginverse=list(Tree=opmet.ainv),na.method.X='include',
+                           workspace=80e6,sparse=~Check)

asreml 3.0-1 (24 October 2012), Library: 3.0hj (15 November 2011), X86_64
     LogLik         S2      DF      wall     cpu
```

```
 -41345.1865      1.0000 10633  21:37:35     2.7
 -41139.6904      1.0000 10633  21:37:37     1.9
 -40965.4398      1.0000 10633  21:37:39     1.9
 -40896.9382      1.0000 10633  21:37:41     1.8
 -40895.3382      1.0000 10633  21:37:42     1.8
 -40895.3359      1.0000 10633  21:37:44     1.9
 -40895.3359      1.0000 10633  21:37:46     1.9
 -40895.3359      1.0000 10633  21:37:48     1.9

Finished on: Mon Jul 15 21:37:48 2013

LogLikelihood Converged

> ss2 <- summary(opmet.tree.asr2,nice=T)$nice
> cov2cor(ss2[[1]])

          FR203_1   FR203_2   FR203_3
FR203_1 1.0000000 0.6515883 0.5282125
FR203_2 0.6515883 1.0000000 0.5322437
FR203_3 0.5282125 0.5322437 1.0000000

> c(opmet.tree.asr1$logl,opmet.tree.asr2$logl)

[1] -40896.02 -40895.34
```

Several comments are required. Firstly, note that we have chosen to use the default starting values for this model. In general we would not recommend this approach for a complex variance model such as the unstructured variance model. The iterative algorithm implemented in `asreml()` is generally very robust for fitting default variance models, as well as some simpler variance models. However, it can be less robust for fitting more complex variance models with default starting values. We are very aware of this issue and we are working on a solution for future releases. For the moment, we recommend a conservative approach, which involves setting up initial values and this approach will be illustrated later in this chapter. Note the use of the `nice` argument to `summary.asreml()` which produces a list of the REML estimates of the variance parameters conformable with the structure of the random and residual model formulae.

The CS variance model is competitive with the unstructured variance model for these data, having similar REML log-likelihoods. In fact it would be the preferred variance model with only two rather than six variance parameters.

6.2.2 Example 3: ARA model

We now present the results of fitting the ARA model to these data. The sequence of variance models is the same as for the full tree model, in that we fit the diagonal, compound symmetric and unstructured variance models to the additive genetic by environment effects. The syntax is slightly more complicated, and illustrates the use of the `str()` variance model function. This is repeatedly used for many of the examples in the remainder of this chapter. We begin with the

diagonal variance model. The syntax and output are:

```
> ###################
> # Now for the ARA model
> # results are identical
> # timings are impressive
> # and the same as single site model
> #
> ###################
> opmet.df$zero <- rep(0,nrow(opmet.df))
> opmet.ram.asr <-
+     asreml(dbh~Expt,random=~str(~Expt:ped(Fcln):zero +
+                                 and(Expt:ped(Fcln),0.5) +
+                                 and(Expt:ped(Mcln),0.5),~diag(Expt):ped(Fcln)) +
+            at(Expt):Replicate + at(Expt):Replicate:Setgroup,
+            rcov=~at(Expt):units,data=opmet.df,
+            ginverse=list(Fcln=opmet.parents.ainv,Mcln=opmet.parents.ainv),
+            na.method.X='include',
+            workspace=80e6,sparse=~Check)

asreml 3.0-1 (24 October 2012), Library: 3.0hj (15 November 2011), X86_64
     LogLik         S2      DF      wall     cpu
 -41747.4422      1.0000 10633  21:37:50     0.2
 -41380.0784      1.0000 10633  21:37:50     0.1
 -41054.0756      1.0000 10633  21:37:50     0.1
 -40911.6016      1.0000 10633  21:37:50     0.1
 -40906.5265      1.0000 10633  21:37:50     0.1
 -40906.5156      1.0000 10633  21:37:50     0.1
 -40906.5156      1.0000 10633  21:37:50     0.1
 -40906.5156      1.0000 10633  21:37:50     0.1

Finished on: Mon Jul 15 21:37:50 2013

LogLikelihood Converged

> ssram <- summary(opmet.ram.asr)$varcomp
> cbind(ssram[grep('.*FR203_1',
+                  dimnames(ssram)[[1]]),'component'],op1.gams[-4,'ARA model'])

              [,1]      [,2]
addvar   124.32925 124.32924
reps      53.91746  53.91746
rep:set   52.48191  52.48191
residual 763.96728 763.96728
```

The reduction in computational burden results in significant savings in computing time per iteration, which is slightly surprising given the simplicity of the variance models and the relatively small data-set. The REML estimates for this model match those obtained from the single site ARA model for FR203_1.

Next we fit the compound symmetric model, but use the `corv()` variance model as we are using `str()`. This variance model is equivalent to fitting a separate main effect and interaction term with default variance models (ie. `idv()`).

```
> ###################
> # now for CS model for ARA
> # fit as corv()
> ###################
> opmet.ram.asr1 <-
+     asreml(dbh~Expt,random=~str(~Expt:ped(Fcln):zero +
+                                 and(Expt:ped(Fcln),0.5) +
+                                 and(Expt:ped(Mcln),0.5),~corv(Expt):ped(Fcln)) +
+            at(Expt):Replicate + at(Expt):Replicate:Setgroup,
+            rcov=~at(Expt):units,data=opmet.df,
+            ginverse=list(Fcln=opmet.parents.ainv,Mcln=opmet.parents.ainv),
+            na.method.X='include',
+            workspace=80e6,sparse=~Check)

asreml 3.0-1 (24 October 2012), Library: 3.0hj (15 November 2011), X86_64
     LogLik          S2      DF      wall     cpu
 -41742.3436      1.0000 10633  21:37:52     0.3
 -41373.4081      1.0000 10633  21:37:52     0.2
 -41045.3416      1.0000 10633  21:37:52     0.2
 -40901.1251      1.0000 10633  21:37:52     0.2
 -40895.8450      1.0000 10633  21:37:53     0.2
 -40895.8313      1.0000 10633  21:37:53     0.2
 -40895.8313      1.0000 10633  21:37:53     0.2
 -40895.8313      1.0000 10633  21:37:53     0.3

Finished on: Mon Jul 15 21:37:53 2013

LogLikelihood Converged

> ssram1 <- summary(opmet.ram.asr1,nice=T)$nice
> ssram1[[1]]

            FR203_1     FR203_2     FR203_3
FR203_1 106.3305737   0.6258487   0.6258487
FR203_2   0.6258487 106.3305737   0.6258487
FR203_3   0.6258487   0.6258487 106.3305737
```

Use of the `nice` argument produces a "nice" format for the REML estimate of $\boldsymbol{G}_a$ from this variance model.

Lastly we fit the unstructured variance model, given by

```
> ###################
> # now for us() model for ARA
> #
```

```
> ###################
> opmet.ram.asr2 <-
+     asreml(dbh~Expt,random=~str(~Expt:ped(Fcln):zero +
+                                 and(Expt:ped(Fcln),0.5) +
+                                 and(Expt:ped(Mcln),0.5),~us(Expt):ped(Fcln)) +
+            at(Expt):Replicate + at(Expt):Replicate:Setgroup,
+            rcov=~at(Expt):units,data=opmet.df,
+            ginverse=list(Fcln=opmet.parents.ainv,Mcln=opmet.parents.ainv),
+            na.method.X='include',
+            workspace=80e6,sparse=~Check)

asreml 3.0-1 (24 October 2012), Library: 3.0hj (15 November 2011), X86_64
     LogLik         S2      DF      wall     cpu
 -41729.0841      1.0000 10633  21:37:55     0.4
 -41364.9199      1.0000 10633  21:37:55     0.2
 -41041.7185      1.0000 10633  21:37:55     0.2
 -40900.2619      1.0000 10633  21:37:56     0.2
 -40895.1761      1.0000 10633  21:37:56     0.2
 -40895.1640      1.0000 10633  21:37:56     0.2
 -40895.1640      1.0000 10633  21:37:56     0.2
 -40895.1640      1.0000 10633  21:37:57     0.2

Finished on: Mon Jul 15 21:37:57 2013

LogLikelihood Converged

> ssram2 <- summary(opmet.ram.asr2,nice=T)$nice
> ssram2[[1]]

          FR203_1  FR203_2  FR203_3
FR203_1 121.82101 68.65761  66.9482
FR203_2  68.65761 91.31408  58.3259
FR203_3  66.94820 58.32590 131.7381

> ss2[[1]]

          FR203_1  FR203_2   FR203_3
FR203_1 121.43532 68.36984  66.80285
FR203_2  68.36984 90.66456  58.16251
FR203_3  66.80285 58.16251 131.71271
```

The REML estimates of $\boldsymbol{G}_a$ from the full tree model and the ARA model are extremely similar, and both confirm the results that the compound symmetric variance model would be the preferred model, with little heterogeneity in variances and in covariances for these data.

6.3 Example 4: Analysis of an clonal MET data-set

In this section we will present an analysis of the clonal MET data-set which comprises three trials, namely FR353_1, FR353_2 and FR353_3. We follow the same approach used for section 6.2 except for brevity we do not present the results for fitting the ARA model.

The first step, after the pre-processing is to set up the parental and individual pedigree data-frames, along with the factor `Check`. There are 793 individuals in the tree pedigree file and 37 parents. There are 215 levels in `Check`.

6.3.1 Example 4: Full tree model

We now present the results for the full tree model and our approach involves modelling both variance structures of the additive and residual genetic by environment effects. This process can be difficult to follow and execute. Problems can arise from fitting *inappropriate* variance models to random model terms which cannot support such models either due to structural or identifiability issues. The latter is a particularly common problem which is often poorly understood, while the latter is also a common problem but appears to be better understood by users of `asreml`. For example, it would be problematic to fit an unstructured variance model to $\boldsymbol{G}_a$ if the REML estimate of one of the variance parameters from the diagonal variance model was set to zero. This is what is meant by an identifiability issue. Secondly, if there were no clones in common between two trials then there is no information to estimate the covariance between trials for the residual genetic effects. This is an example of a structural issue. Given the excellent connectivity only identifiability issues may arise when fitting a model such as the unstructured variance model.

The syntax and output for the fit of the diagonal variance model to both $\boldsymbol{G}_a$ and $\boldsymbol{G}_e$ are

```
> ###################################
> # now we fit the tree model
> #
> #
> # All three trials are Set = 1 ie dont fit setgroup
> # all are IB, all are Plottype = 1
> # matches ok!
> ##############################
> clonemet.tree.asr <- asreml(dbh~Expt,random=~diag(Expt):ped(Tree) +
+                             at(Expt):Replicate + at(Expt):Replicate:Iblk +
+                             diag(Expt):ide(Tree),
+                             rcov=~at(Expt):units,data=clonemet.df,
+                             ginverse=list(Tree=clonemet.ainv),
+                             na.method.X='include',
+                             workspace=80e6,sparse=~Check)

asreml 3.0-1 (24 October 2012), Library: 3.0hj (15 November 2011), X86_64
     LogLik          S2      DF      wall     cpu
 -27436.9249      1.0000  6848  21:37:58     0.3 (1 restrained)
```

```
 -27318.9511      1.0000  6848  21:37:58     0.2
 -27218.1813      1.0000  6848  21:37:59     0.2
 -27176.1995      1.0000  6848  21:37:59     0.2
 -27174.2062      1.0000  6848  21:37:59     0.2
 -27174.1121      1.0000  6848  21:37:59     0.2
 -27174.1097      1.0000  6848  21:37:59     0.2
 -27174.1097      1.0000  6848  21:37:59     0.2
 -27174.1097      1.0000  6848  21:38:00     0.2

Finished on: Mon Jul 15 21:38:00 2013

LogLikelihood Converged

> ssclone <- summary(clonemet.tree.asr)$varcomp
> cbind(ssclone[grep('.*FR353_1',dimnames(ssclone)[[1]]),'component'],
+       clone1.gams[-5,'Tree model'])

                [,1]       [,2]
addvar    256.270200 256.269906
reps        3.927676   3.927678
rep:ib     81.639557  81.639544
nonaddvar 166.868193 166.868374
residual  790.181943 790.181994

> niceclone <- summary(clonemet.tree.asr,nice=T)$nice
> cbind(niceclone[["Expt:ped(Tree)"]],niceclone[["Expt:ide(Tree)"]],
+        unlist(niceclone[9:11]))

                      [,1]      [,2]      [,3]
Expt.FR353_1.var 256.27020 166.86819  790.1819
Expt.FR353_2.var 408.92243  22.75462 1052.0171
Expt.FR353_3.var  90.46845  94.15763  692.1088

>
```

The results agree with the single site analysis for trial FR353_1. There is evidence of variance heterogeneity for both additive and residual genetic by environment effects and we note that the REML estimate of residual genetic variance for the FR353_2 trial is small and the REML estimates of additive and residual genetic variance for the FR353_3 trial are small relative to the residual error variance. This may impede our ability to fit more complex variance models, ie. identifiability issues.

We now fit the compound symmetric model to both $\boldsymbol{G}_a$ and $\boldsymbol{G}_e$. This model is unlikely to provide a good fit, given the apparent heterogeneity observed from the fit of the diagonal variance models. The syntax and output are:

```
> ################
> # ok tree model again but A+A:E
```

```
> # methinks it wont work that well!
> # to save time do it at the ped and ide levels
> # in the one model
> #################
> clonemet.tree.asr1 <- asreml(dbh~Expt,random=~ped(Tree) + Expt:ped(Tree) +
+                              at(Expt):Replicate + at(Expt):Replicate:Iblk +
+                              ide(Tree) + Expt:ide(Tree),
+                              rcov=~at(Expt):units,data=clonemet.df,
+                              ginverse=list(Tree=clonemet.ainv),
+                              na.method.X='include',
+                              workspace=80e6,sparse=~Check)

asreml 3.0-1 (24 October 2012), Library: 3.0hj (15 November 2011), X86_64
     LogLik          S2      DF      wall     cpu
 -27240.3350      1.0000  6848  21:38:01     0.6 (1 restrained)
 -27173.8404      1.0000  6848  21:38:01     0.3
 -27120.5228      1.0000  6848  21:38:02     0.4
 -27101.6737      1.0000  6848  21:38:02     0.4
 -27101.4583      1.0000  6848  21:38:03     0.3
 -27101.4581      1.0000  6848  21:38:03     0.3
 -27101.4581      1.0000  6848  21:38:03     0.4

Finished on: Mon Jul 15 21:38:03 2013

LogLikelihood Converged

> ssclone1 <- summary(clonemet.tree.asr1)$varcomp
> niceclone1 <- summary(clonemet.tree.asr1,nice=T)$nice
> metcs.gams <-
+     rbind(cbind(niceclone1[["ped(Tree)"]],niceclone1[["ide(Tree)"]]),
+           cbind(niceclone1[["Expt:ped(Tree)"]],niceclone1[["Expt:ide(Tree)"]]))
> dimnames(metcs.gams) <- list(c('main','inter'),c('ped','ide'))
> metcs.gams

            ped     ide
main  149.57258 68.8601
inter  87.70263 34.5290
```

We now consider fitting the unstructured variance model to both $\boldsymbol{G}_a$ and $\boldsymbol{G}_e$.

```
> ###################
> # tree model with us for both
> #
> ###################
> clonemet.tree.asr2 <- asreml(dbh~Expt,random=~us(Expt):ped(Tree) +
+                              at(Expt):Replicate + at(Expt):Replicate:Iblk +
+                              us(Expt):ide(Tree),
+                              rcov=~at(Expt):units,data=clonemet.df,
+                              ginverse=list(Tree=clonemet.ainv),
```

```
+                              na.method.X='include',
+                              workspace=80e6,sparse=~Check)

asreml 3.0-1 (24 October 2012), Library: 3.0hj (15 November 2011), X86_64
     LogLik          S2      DF      wall     cpu
 -27276.0690      1.0000  6848  21:38:04     0.4 (1 restrained)
 US matrix updates modified 1 times to remain positive definite.
 -27178.0595      1.0000  6848  21:38:05     0.3 (6 restrained)
 US matrix updates modified 1 times to remain positive definite.
 -27102.0772      1.0000  6848  21:38:05     0.3 (6 restrained)
 US matrix updates modified 1 times to remain positive definite.
 -27073.5542      1.0000  6848  21:38:05     0.3 (6 restrained)
 US matrix updates modified 1 times to remain positive definite.
 -27072.7244      1.0000  6848  21:38:06     0.3 (6 restrained)
 US matrix updates modified 1 times to remain positive definite.
 -27072.5234      1.0000  6848  21:38:06     0.3 (6 restrained)
 US matrix updates modified 1 times to remain positive definite.
 -27072.3326      1.0000  6848  21:38:06     0.3 (6 restrained)
 US matrix updates modified 1 times to remain positive definite.
 -27072.1468      1.0000  6848  21:38:06     0.2 (6 restrained)
 US matrix updates modified 1 times to remain positive definite.
 -27071.9658      1.0000  6848  21:38:07     0.3 (6 restrained)
 US matrix updates modified 1 times to remain positive definite.
 -27071.7895      1.0000  6848  21:38:07     0.3 (6 restrained)
 -27071.6178      1.0000  6848  21:38:07     0.3 (6 restrained)
 -27071.4506      1.0000  6848  21:38:07     0.3 (6 restrained)
 -27071.2879      1.0000  6848  21:38:08     0.3 (6 restrained)
US variance structures were modified in 12 instances to make them positive definite

Finished on: Mon Jul 15 21:38:08 2013

LogLikelihood not converged
```

which fails to converge using the default starting values. Although these models may not be easy to fit to these data, given the potential identifiability issues, we will attempt to fit the same model but using user-supplied starting values. This is achieved in the following syntax, which contains some *challenging* concepts and R programming.

```
> ####################
> # fails to converge....
> #
> ####################
> ne <- length(levels(clonemet.df$Expt))
> clonemet.tree.sv <- asreml(dbh~Expt,random=~us(Expt):ped(Tree) +
+                              at(Expt):Replicate + at(Expt):Replicate:Iblk +
+                              us(Expt):ide(Tree),
+                              rcov=~at(Expt):units,data=clonemet.df,
+                              ginverse=list(Tree=clonemet.ainv),
+                              na.method.X='include',start.values=T,
```

```
+                              workspace=80e6,sparse=~Check)
> temp <- clonemet.tree.sv$gammas.table
> temp[grep('.*Tree',temp$Gamma,invert=T),'Value'] <-
+     ssclone1[grep('.*Tree',dimnames(ssclone1)[[1]],invert=T),'component']
> ###############
> # ped first
> rhoga <- metcs.gams['main','ped']/(metcs.gams['main','ped'] +
+                                    metcs.gams['inter','ped'])
> Ga <- matrix(rhoga,ne,ne)
> diag(Ga) <- rep(1,ne)
> Gase <- diag(sqrt(niceclone[['Expt:ped(Tree)']]))
> Ga <- Gase%*%Ga%*%Gase
> temp[grep('.*ped\\(Tree\\)',temp$Gamma,invert=F),'Value'] <- Ga[!lower.tri(Ga)]
> ##############
> # ide next
> rhogi <- metcs.gams['main','ide']/(metcs.gams['main','ide'] +
+                                    metcs.gams['inter','ide'])
> Ge <- matrix(rhoga,ne,ne)
> diag(Ge) <- rep(1,ne)
> Gese <- diag(sqrt(niceclone[['Expt:ide(Tree)']]))
> Ge <- Gese%*%Ge%*%Gese
> temp[grep('.*ide\\(Tree\\)',temp$Gamma,invert=F),'Value'] <- Ge[!lower.tri(Ge)]
> ##############
> # tried it again with new starting values
> # but unsuccessful
> # suspect it is ide() term causing the trouble
> # numerous updates no go
> # both terms within space???
> #################
> clonemet.tree.asr1 <- asreml(dbh~Expt,random=~us(Expt):ped(Tree) +
+                              at(Expt):Replicate + at(Expt):Replicate:Iblk +
+                              us(Expt):ide(Tree),
+                              rcov=~at(Expt):units,data=clonemet.df,
+                              ginverse=list(Tree=clonemet.ainv),
+                              na.method.X='include',R.param=temp,G.param=temp,
+                              workspace=80e6,sparse=~Check)

asreml 3.0-1 (24 October 2012), Library: 3.0hj (15 November 2011), X86_64 US matrix updates modified

     LogLik          S2      DF      wall     cpu
 -27085.0690      1.0000  6848  21:38:09     0.4 (6 restrained)
 US matrix updates modified 1 times to remain positive definite.
 -27083.6992      1.0000  6848  21:38:09     0.2 (1 restrained)
 US matrix updates modified 2 times to remain positive definite.
 -27123.8881      1.0000  6848  21:38:10     0.2 (10 restrained)
 US matrix updates modified 2 times to remain positive definite.
 -27114.2595      1.0000  6848  21:38:10     0.3 (10 restrained)
 US matrix updates modified 2 times to remain positive definite.
 -27107.5859      1.0000  6848  21:38:10     0.2 (10 restrained)
 -27102.4295      1.0000  6848  21:38:10     0.3 (10 restrained)
```

```
 -27098.3357      1.0000  6848  21:38:11     0.3 (10 restrained)
 -27095.0120      1.0000  6848  21:38:11     0.3 (11 restrained)
 -27092.2634      1.0000  6848  21:38:11     0.3 (11 restrained)
 -27089.9552      1.0000  6848  21:38:12     0.3 (11 restrained)
 -27087.9978      1.0000  6848  21:38:12     0.3 (11 restrained)
 -27086.3127      1.0000  6848  21:38:12     0.3 (12 restrained)
 -27084.8483      1.0000  6848  21:38:12     0.2 (10 restrained)
US variance structures were modified in 25 instances to make them positive definite

Finished on: Mon Jul 15 21:38:12 2013

LogLikelihood not converged

> clonemet.tree.asr1 <- update(clonemet.tree.asr1)

asreml 3.0-1 (24 October 2012), Library: 3.0hj (15 November 2011), X86_64 US matrix updates modified

     LogLik         S2      DF      wall     cpu
 -27083.5611      1.0000  6848  21:38:13     0.4 (6 restrained)
 US matrix updates modified 1 times to remain positive definite.
 -27076.4341      1.0000  6848  21:38:14     0.3 (6 restrained)
 US matrix updates modified 2 times to remain positive definite.
 -27072.5578      1.0000  6848  21:38:14     0.3 (12 restrained)
 US matrix updates modified 2 times to remain positive definite.
 -27072.4231      1.0000  6848  21:38:14     0.3 (12 restrained)
 US matrix updates modified 2 times to remain positive definite.
 -27072.2951      1.0000  6848  21:38:15     0.3 (12 restrained)
 -27072.1728      1.0000  6848  21:38:15     0.3 (12 restrained)
 -27072.0559      1.0000  6848  21:38:15     0.3 (12 restrained)
 -27071.9439      1.0000  6848  21:38:15     0.3 (12 restrained)
 -27071.8365      1.0000  6848  21:38:16     0.3 (12 restrained)
 -27071.7336      1.0000  6848  21:38:16     0.3 (12 restrained)
 -27071.6346      1.0000  6848  21:38:16     0.3 (12 restrained)
 -27071.5395      1.0000  6848  21:38:16     0.3 (6 restrained)
 -27071.1799      1.0000  6848  21:38:17     0.3 (6 restrained)
US variance structures were modified in 22 instances to make them positive definite

Finished on: Mon Jul 15 21:38:17 2013

LogLikelihood not converged

> cov2cor(summary(clonemet.tree.asr1,nice=T)$nice[[1]])

          FR353_1   FR353_2   FR353_3
FR353_1 1.0000000 0.7925409 0.6493189
FR353_2 0.7925409 1.0000000 0.5113180
FR353_3 0.6493189 0.5113180 1.0000000

> cov2cor(summary(clonemet.tree.asr1,nice=T)$nice[[8]])
```

```
          FR353_1   FR353_2   FR353_3
FR353_1 1.0000000 0.9066459 0.7734235
FR353_2 0.9066459 1.0000000 0.4655473
FR353_3 0.7734235 0.4655473 1.0000000
```

This chunk of the syntax uses the `start.values` argument of `asreml()` to provide starting values to subsequent calls to `asreml()`. The `grep()` function is an excellent, albeit abstract function which we commonly use for a range of sub-setting problems. We note, however, that our efforts are in vain, as repeated calls to `asreml()` fail to converge. Each update appears to produce illegal parameter estimates with either 6 or 12 parameters constrained in the call to `update.asreml()`.

(In frustration!) we now proceed to fit FA models to both $\boldsymbol{G}_a$ and $\boldsymbol{G}_e$. We have found that these models can often be fitted to data-sets where we are unable to fit unstructured variance models, even though the dimensions of the variance matrices involved are modest.

We find that this model converges even using default starting values.

```
> ####################
> # perhaps try fa(ped) + fa(ide)
> # realise this is not the same model
> # but it may converge
> # even with naive starting values
> #
> ####################
> clonemet.tree.asr1 <- asreml(dbh~Expt,random=~fa(Expt):ped(Tree) +
+                              at(Expt):Replicate + at(Expt):Replicate:Iblk +
+                              fa(Expt):ide(Tree),
+                              rcov=~at(Expt):units,data=clonemet.df,
+                              ginverse=list(Tree=clonemet.ainv),
+                              na.method.X='include',
+                              workspace=80e6,sparse=~Check)

asreml 3.0-1 (24 October 2012), Library: 3.0hj (15 November 2011), X86_64
     LogLik         S2      DF      wall     cpu
 -28291.2890      1.0000  6848  21:38:18     0.5 (4 restrained)
 -27416.4278      1.0000  6848  21:38:18     0.3 (9 restrained)
 -27302.0767      1.0000  6848  21:38:18     0.3 (8 restrained)
 -27243.2684      1.0000  6848  21:38:19     0.5 (7 restrained)
 -27207.7198      1.0000  6848  21:38:19     0.3 (7 restrained)
 -27165.7390      1.0000  6848  21:38:20     0.4 (6 restrained)
 -27120.9238      1.0000  6848  21:38:20     0.3
 -27080.0452      1.0000  6848  21:38:20     0.3
 -27070.2231      1.0000  6848  21:38:20     0.2
 -27069.7770      1.0000  6848  21:38:21     0.3
 -27069.7718      1.0000  6848  21:38:21     0.2
 -27069.7710      1.0000  6848  21:38:21     0.3
 -27069.7709      1.0000  6848  21:38:21     0.2
```

```
Finished on: Mon Jul 15 21:38:21 2013

LogLikelihood not converged

> clonemet.tree.asr1 <- update(clonemet.tree.asr1)

asreml 3.0-1 (24 October 2012), Library: 3.0hj (15 November 2011), X86_64
     LogLik          S2      DF      wall     cpu
 -27069.7709      1.0000  6848  21:38:22     0.4
 -27069.7709      1.0000  6848  21:38:23     0.3
 -27069.7709      1.0000  6848  21:38:23     0.3
 -27069.7709      1.0000  6848  21:38:23     0.3

Finished on: Mon Jul 15 21:38:23 2013

LogLikelihood Converged

> niceclone1 <- summary(clonemet.tree.asr1,nice=T)$nice
> ped.lam.clone1 <- matrix(niceclone1[["fa(Expt):ped(Tree)"]][4:6],3,1)
> ped.psi.clone1 <- diag(niceclone1[["fa(Expt):ped(Tree)"]][1:3])
> Ga.clone <- ped.lam.clone1%*%t(ped.lam.clone1) + ped.psi.clone1
> cov2cor(Ga.clone)

          [,1]      [,2]      [,3]
[1,] 1.0000000 0.9290788 0.5815777
[2,] 0.9290788 1.0000000 0.5403315
[3,] 0.5815777 0.5403315 1.0000000

> ide.lam.clone1 <- matrix(niceclone1[["fa(Expt):ide(Tree)"]][4:6],3,1)
> ide.psi.clone1 <- diag(niceclone1[["fa(Expt):ide(Tree)"]][1:3])
> Ge.clone <- ide.lam.clone1%*%t(ide.lam.clone1) + ide.psi.clone1
> cov2cor(Ge.clone)

          [,1]      [,2]      [,3]
[1,] 1.0000000 0.8139393 0.8260097
[2,] 0.8139393 1.0000000 0.6723218
[3,] 0.8260097 0.6723218 1.0000000
```

One feature which we will address in the next release will be to return the REML estimate of $\boldsymbol{G}_s, s = a, e$ when using the `nice` argument of `summary.asreml()`, rather than just the loadings and specific variances in a vector (loadings last).

6.4 Example 5: Analysis of a combined OP/CP/clonal MET data-set

We conclude this chapter with the analysis of the full set of trials considered in this course. This is an OP/CP/clonal MET data-set, but, by our standards, only relatively small and with good parental and clonal connectivity. We will consider both the full tree and ARA models and will

fit a range of variance models to both $\boldsymbol{G}_a$ and $\boldsymbol{G}_e$.

Again much of the pre-processing is the same as before and hence is not presented.

```
> ##################
> # this is the script to run the single site OP analyses
> # we will read in the data and ped files and then
> # fit tree and RAM models
> #
> ########################
> require(asreml)
> source("C:/courses/vancouver/scripts/pedtrim.R")
> source("C:/courses/vancouver/scripts/nfa-fns.R")
> ################
> # first read in the data and ped files
> #
> # set up control trees
> # 8 control trees in pedigree and data
> #################
> ws.df <- read.table("c:/courses/vancouver/scripts/wsdata.csv",header=T,sep=',')
> ws.df <- ws.df[,-c(1,2)]
> ws.df <- ws.df[order(ws.df$Expt),]
> wsped.df <- read.table("c:/courses/vancouver/scripts/wspedtree.csv",
+                        header=T,sep=',')
> wsped.df <- wsped.df[,-1]
> head(wsped.df)
> nrow(wsped.df) # 12714
> control.numbers <- c(111020,111039,111111,111227,111228,111266,111268,111270,111525,
+                      111526,
+                      111527,111602,111606,111607,111613,111614,111616,111618,111619,
+                      111626,
+                      111628,111630,111646,111647,111652,111674,111676,111678,111713,
+                      111791,
+                      111792,111854,111870,111871,111872,111930,111991,111992,111993,
+                      111997,
+                      111998)
> nrow(subset(wsped.df,is.element(Tree,control.numbers))) #8
> ##################
> # select the three clonal sites
> # and use pedtrim to reduce pedigrees to be specific to each trial
> # 12714 in full pedigree for 6 trials
> #
> ##################
> met.df <- ws.df
> temp.ped <- data.ped.trim(pedfile=wsped.df,data=met.df,parent=TRUE)
> met.tree.df <- temp.ped$tree.ped #[1] 12714
> met.parent.df <- temp.ped$parent.ped #[1] 302
> ####################
> # Set up the Check factor
```

```
> # as I said in the course notes you can drclone the data records for the Checks
> # or leave them in and fit the Check factor as a fixed effect
> # the end result is the same
> # the latter approach is a neat trick which we choose to demonstrate
> # here
> # set up the control factor which is
> # based on Fcln and Mcln numbers
> # >111000 and <111999
> #
> # this creates a factor with
> # 1235 levels
> ######################
> temp <- rep(1,nrow(met.df))
> temp[is.element(met.df$Fcln,control.numbers)] <-
+     met.df$Tree[is.element(met.df$Fcln,control.numbers)]
> length(unique(temp))
> temp[is.element(met.df$Mcln,control.numbers)] <-
+     met.df$Tree[is.element(met.df$Mcln,control.numbers)]
> t1 <- subset(met.df,is.element(met.df$Mcln,control.numbers))$Tree
> t2 <- subset(met.df,is.element(met.df$Fcln,control.numbers))$Tree
> c(sum(is.element(t1,t2)),sum(is.element(t2,t1)))
> c(length(unique(temp)),length(t1)+length(t2))
> met.df$Check <- factor(temp)
> c(nrow(subset(met.df,is.element(met.df$Fcln,control.numbers)))+
+   nrow(subset(met.df,is.element(met.df$Mcln,control.numbers))),
+   length(levels(met.df$Check))) # [1] 1234 1235
> ############################
> # get the bits for the RAM approach
> # we use the diagonal of A^{-1} for trees in the trial
> # first get those animals which are not used as parents
> # we will do met as an example and leave the other
> # two trials as class exercises
> #
> ############################
> parents.met <- met.parent.df$Tree
> nonparents.met <- met.tree.df$Tree
> nonparents.met <-
+     nonparents.met[!is.element(nonparents.met,parents.met)]
> c(length(parents.met),length(nonparents.met))
> temp <- asreml.Ainverse(met.parent.df)
> table(temp$inbreeding)
> met.parents.ainv <- temp$ginv
> temp <- asreml.Ainverse(met.tree.df)
> table(temp$inbreeding)
> met.ainv <- temp$ginv
```

We note that there are 12714 individuals in the full pedigree but only 302 parents. This will result in substantial reductions in computing times between the full tree and ARA model, without forward selections, but there will be less improvement for the ARA model with forward

selections. Before we commence our analyses, we will create some useful objects and some additional columns in the data frame. Firstly, we have found that it is useful to create objects which delineate which trials require which terms in the fixed and random model formulae. Here, we define objects to allow fitting blocks within replicates and sets with replicates. The syntax is presented below.

```
> ############
> # create some identifiers for the analysis to determine
> # which effects to fit to which trial
> #
> ############
> ii <- with(met.df,tapply(Iblk,Expt,function(x) length(unique(x))))
> met.pblk <- names(ii)[ii>1]
> ii <- with(met.df,tapply(Setgroup,Expt,function(x) length(unique(x))))
> met.pset <- names(ii)[ii>1]
> ii <- with(met.df,tapply(Plot,Expt,function(x) length(unique(x))))
> met.pplt <- names(ii)[ii>1]
> ii <- with(met.df,tapply(Clone,Expt,function(x) length(unique(x))))
> met.pclon <- names(ii)[ii>1]
```

We now create a copy of the `Tree` factor and the `Expt` factor which are set to `NA` for all data in non-clonal trials.

```
> ################
> # create the Tree factor for the clonal trials
> # with NAs elsewhere!!!
> # and also a copy of the Expt factor for the clonal trials
> #
> ################
> temp <- rep(NA,nrow(met.df))
> temp[is.element(met.df$Expt,met.pclon)] <-
+     as.character(met.df$Tree)[is.element(met.df$Expt,met.pclon)]
> met.df$Tclone <- factor(temp)
> length(levels(met.df$Tclone)) # 1750

[1] 756

> met.df$TExpt <- met.df$Expt
> met.df$TExpt[!is.element(met.df$Expt,met.pclon)] <- NA
> met.df$TExpt <- factor(as.character(met.df$TExpt))
```

6.4.1 Example 5: Full tree model

We commence by fitting diagonal variance models to both $\boldsymbol{G}_a$ and $\boldsymbol{G}_e$ but note that the latter can only be fitted to the three clonal trials, necessitating the use of the two new columns created in the above code chunk.

```
> ##########################
> # end of starting section
> # now for Tree model
> # diag for both, but must fit the
> # ide carefully to only clonal trials
> # checks with Clone Tree model
> # for 353_1
> ##########################
> met.tree.asr <- asreml(dbh~Expt,random=~diag(Expt):ped(Tree) +
+                        at(Expt):Replicate + at(Expt,met.pblk):Replicate:Iblk +
+                        at(Expt,met.pset):Replicate:Setgroup +
+                        diag(TExpt):Tclone,
+                        rcov=~at(Expt):units,data=met.df,
+                        ginverse=list(Tree=met.ainv), na.method.X='include',
+                        workspace=80e6,sparse=~Check)

asreml 3.0-1 (24 October 2012), Library: 3.0hj (15 November 2011), X86_64
     LogLik         S2      DF      wall     cpu
 -68848.9250      1.0000 17481  21:38:28     2.2 (1 restrained)
 -68499.2881      1.0000 17481  21:38:30     1.6
 -68202.9196      1.0000 17481  21:38:32     1.5
 -68084.2351      1.0000 17481  21:38:33     1.6
 -68080.8607      1.0000 17481  21:38:35     1.5
 -68080.8139      1.0000 17481  21:38:36     1.5
 -68080.8131      1.0000 17481  21:38:38     1.6
 -68080.8131      1.0000 17481  21:38:39     1.6

Finished on: Mon Jul 15 21:38:40 2013

LogLikelihood Converged

> ssmet <- summary(met.tree.asr)$varcomp
> cbind(ssmet[grep('.*FR353_1',dimnames(ssmet)[[1]]),'component'],
+       clone1.gams[-5,'Tree model'])

                [,1]       [,2]
addvar    256.269942 256.269906
reps        3.927678   3.927678
rep:ib     81.639542  81.639544
nonaddvar 166.868335 166.868374
residual  790.181945 790.181994
```

We now attempt to fit unstructured variance matrices, using default starting values and find that the fit fails to converge. This is not surprising given the results of the preceding sections, particularly our failed attempts to fit this model to the clonal MET data-set.

```
> ######################
> # ok lets go for us + us
> # not what I would normally do...
```

```
> # fails as expected
> # 12 secs for each iteration
> ######################
> met.tree.asr1 <- asreml(dbh~Expt,random=~us(Expt):ped(Tree) +
+                         at(Expt):Replicate + at(Expt,met.pblk):Replicate:Iblk +
+                         at(Expt,met.pset):Replicate:Setgroup +
+                         us(TExpt):Tclone,
+                         rcov=~at(Expt):units,data=met.df,
+                         ginverse=list(Tree=met.ainv), na.method.X='include',
+                         workspace=80e6,sparse=~Check)

asreml 3.0-1 (24 October 2012), Library: 3.0hj (15 November 2011), X86_64 US matrix updates modified

     LogLik          S2      DF      wall     cpu
 -68586.9488      1.0000 17481  21:38:56    14.2 (1 restrained)
 US matrix updates modified 1 times to remain positive definite.
 -68295.4455      1.0000 17481  21:39:07    11.4 (21 restrained)
 US matrix updates modified 1 times to remain positive definite.
 -68065.5357      1.0000 17481  21:39:18    11.3 (21 restrained)
 US matrix updates modified 2 times to remain positive definite.
 -67980.7178      1.0000 17481  21:39:30    11.6 (27 restrained)
 US matrix updates modified 2 times to remain positive definite.
 -67979.0457      1.0000 17481  21:39:41    11.1 (27 restrained)
 US matrix updates modified 2 times to remain positive definite.
 -67978.9544      1.0000 17481  21:39:53    11.7 (27 restrained)
 -67978.8731      1.0000 17481  21:40:05    11.6 (27 restrained)
 -67978.7984      1.0000 17481  21:40:16    11.5 (21 restrained)
 -67978.6456      1.0000 17481  21:40:28    11.6 (21 restrained)
 -67978.5946      1.0000 17481  21:40:39    11.5 (21 restrained)
 -67978.5444      1.0000 17481  21:40:51    11.5 (21 restrained)
 -67978.4945      1.0000 17481  21:41:02    11.3 (21 restrained)
 -67978.4449      1.0000 17481  21:41:14    11.7 (21 restrained)
US variance structures were modified in 17 instances to make them positive definite

Finished on: Mon Jul 15 21:41:14 2013

LogLikelihood not converged

> met.tree.asr1  <- update(met.tree.asr1)

asreml 3.0-1 (24 October 2012), Library: 3.0hj (15 November 2011), X86_64 US matrix updates modified

     LogLik          S2      DF      wall     cpu
 -67978.3957      1.0000 17481  21:41:30    14.5 (21 restrained)
 US matrix updates modified 1 times to remain positive definite.
 -67978.3469      1.0000 17481  21:41:42    12.0 (21 restrained)
 US matrix updates modified 1 times to remain positive definite.
 -67978.2985      1.0000 17481  21:41:54    11.6 (21 restrained)
 US matrix updates modified 1 times to remain positive definite.
```

```
 -67978.2503      1.0000 17481  21:42:06    11.8 (21 restrained)
 US matrix updates modified 1 times to remain positive definite.
 -67978.2023      1.0000 17481  21:42:18    12.0 (21 restrained)
 US matrix updates modified 1 times to remain positive definite.
 -67978.1547      1.0000 17481  21:42:29    11.7 (21 restrained)
 US matrix updates modified 1 times to remain positive definite.
 -67978.1073      1.0000 17481  21:42:41    11.8 (21 restrained)
 US matrix updates modified 1 times to remain positive definite.
 -67978.0603      1.0000 17481  21:42:53    11.8 (21 restrained)
 US matrix updates modified 1 times to remain positive definite.
 -67978.0135      1.0000 17481  21:43:05    12.1 (21 restrained)
 -67977.9670      1.0000 17481  21:43:17    12.3 (21 restrained)
 -67977.9209      1.0000 17481  21:43:30    12.3 (21 restrained)
 -67977.8750      1.0000 17481  21:43:42    12.1 (21 restrained)
 -67977.8294      1.0000 17481  21:43:53    11.8 (21 restrained)
US variance structures were modified in 13 instances to make them positive definite

Finished on: Mon Jul 15 21:43:54 2013

LogLikelihood not converged
```

For the remainder of this section we therefore focus on models using FA variance models for both $\boldsymbol{G}_a$ and $\boldsymbol{G}_e$. We commence with FA(1) variance models for both matrices, using default starting values:

```
> ######################
> # ok lets go for fa1 + fa1
> # with default starting values (a risk!!!)
> # convergence achieved with one update
> # dont you love ASREML!!
> ######################
> met.tree.asr1 <- asreml(dbh~Expt,random=~fa(Expt):ped(Tree) +
+                         at(Expt):Replicate + at(Expt,met.pblk):Replicate:Iblk +
+                         at(Expt,met.pset):Replicate:Setgroup +
+                         fa(TExpt):Tclone,
+                         rcov=~at(Expt):units,data=met.df,
+                         ginverse=list(Tree=met.ainv), na.method.X='include',
+                         workspace=80e6,sparse=~Check)

asreml 3.0-1 (24 October 2012), Library: 3.0hj (15 November 2011), X86_64
     LogLik         S2      DF      wall     cpu
 -70272.9766      1.0000 17481  21:44:27    31.0 (7 restrained)
 -68836.4116      1.0000 17481  21:44:35     8.5 (12 restrained)
 -68430.2518      1.0000 17481  21:44:44     8.9 (11 restrained)
 -68207.7999      1.0000 17481  21:45:15    31.0 (10 restrained)
 -68105.3992      1.0000 17481  21:45:24     9.1 (10 restrained)
 -68048.3205      1.0000 17481  21:45:33     8.6 (10 restrained)
 -68020.1198      1.0000 17481  21:45:41     8.8 (10 restrained)
 -67997.2970      1.0000 17481  21:45:50     8.6 (1 restrained)
```

```
 -67973.8387      1.0000 17481  21:45:59     9.4 (1 restrained)
 -67968.0745      1.0000 17481  21:46:08     8.7 (1 restrained)
 -67965.8182      1.0000 17481  21:46:32    23.6
 -67963.8784      1.0000 17481  21:46:39     7.3
 -67963.5648      1.0000 17481  21:46:46     7.3

Finished on: Mon Jul 15 21:46:47 2013

LogLikelihood not converged

> met.tree.asr1  <- update(met.tree.asr1)

asreml 3.0-1 (24 October 2012), Library: 3.0hj (15 November 2011), X86_64
     LogLik          S2      DF      wall     cpu
 -67963.5486      1.0000 17481  21:47:12    23.0
 -67963.5485      1.0000 17481  21:47:19     7.5
 -67963.5483      1.0000 17481  21:47:27     7.3
 -67963.5478      1.0000 17481  21:47:34     7.4
 -67963.5476      1.0000 17481  21:47:41     7.3

Finished on: Mon Jul 15 21:47:42 2013

LogLikelihood Converged
                                    Change(%)
fa(TExpt):Tclone!TExpt.FR353_2.var      1.27

> ssmet1 <- summary(met.tree.asr1)$varcomp
```

We do not present summaries for this model, as we wish to demonstrate our approach to model building for MET data-sets. This generally involves fitting a sequence of FA variance models for $k = 1, 2, \ldots, k_m$ where k_m is the maximal FA model which we either can fit or choose to fit. For these data, since $t = 6$, $k_m = 3$, but this has the same number of parameters as the unstructured variance model.

The results for fitting the FA(2) and FA(1) variance models to $\boldsymbol{G}_a$ and $\boldsymbol{G}_e$ respectively is now presented. We note that we do not use default starting values, but choose to use starting values taken from the previous FA(1) model fit. The only variance model which is a different variance model for the random and residual model terms is the variance model for $\boldsymbol{G}_a$. Hence we use the REML estimates from the previous fit as "sensible" starting values for all other variance parameters other the variance parameters associated with $\boldsymbol{G}_a$. For these we do the following. We use the REML estimates of the FA(1) parameters from the previous fit as starting values for factor 1 loadings and the specific variances. For the factor 2 loadings we use the default starting values, noting that `asreml()` sets and constrains the first value of this vector to zero.

```
> ######################
> # ok lets go for fa2 + fa1
> # with fa1 starting values
```

```
> # had to use my own update function
> # and change the maxit = 10 for the first call to asreml()
> # research problem!
> ######################
> met.tree.sv <- asreml(dbh~Expt,random=~fa(Expt,2):ped(Tree) +
+                       at(Expt):Replicate + at(Expt,met.pblk):Replicate:Iblk +
+                       at(Expt,met.pset):Replicate:Setgroup +
+                       fa(TExpt):Tclone,
+                       rcov=~at(Expt):units,data=met.df,start.values=T,
+                       ginverse=list(Tree=met.ainv), na.method.X='include',
+                       workspace=80e6,sparse=~Check)
> temp <- met.tree.sv$gammas.table
> temp[grep('.*Tree',temp$Gamma,invert=T),'Value'] <-
+     ssmet1[grep('.*Tree',dimnames(ssmet1)[[1]],invert=T),'component']
> nemet <- length(levels(met.df$Expt))
> fa2.svs <- c(summary(met.tree.asr1,nice=T)$nice[["fa(Expt):ped(Tree)"]],
+              c(0,rep(.1,nemet-1)))
> temp[grep('.*Tree',temp$Gamma,invert=F),'Value'] <- fa2.svs
> met.tree.asr2 <- asreml(dbh~Expt,random=~fa(Expt,2):ped(Tree) +
+                       at(Expt):Replicate + at(Expt,met.pblk):Replicate:Iblk +
+                       at(Expt,met.pset):Replicate:Setgroup +
+                       fa(TExpt):Tclone,R.param=temp,G.param=temp,
+                       rcov=~at(Expt):units,data=met.df,maxit=10,
+                       ginverse=list(Tree=met.ainv), na.method.X='include',
+                       workspace=80e6,sparse=~Check)

asreml 3.0-1 (24 October 2012), Library: 3.0hj (15 November 2011), X86_64
     LogLik         S2      DF      wall     cpu
 -67963.5478      1.0000 17481  21:48:16    30.8
Logliklihood decreased to  -68564.65 - trying again with reduced updates
 -67967.2861      1.0000 17481  21:48:41    25.8
 -67966.4024      1.0000 17481  21:48:55    13.3
 -67964.1225      1.0000 17481  21:49:09    13.8
 -67965.0963      1.0000 17481  21:49:21    12.6 (1 restrained)
 -67963.4469      1.0000 17481  21:49:34    13.0 (1 restrained)
 -67960.9037      1.0000 17481  21:49:48    13.4 (1 restrained)
 -67960.2908      1.0000 17481  21:50:01    13.6 (1 restrained)
 -67960.1790      1.0000 17481  21:50:22    21.0
 -67960.1386      1.0000 17481  21:50:32    10.0

Finished on: Mon Jul 15 21:50:33 2013

LogLikelihood not converged

> temp.asr  <- update.fa(met.tree.asr2,startsearch=F,endsearch=F,midsearch=T,
+                       convtest=F)

asreml 3.0-1 (24 October 2012), Library: 3.0hj (15 November 2011), X86_64
     LogLik         S2      DF      wall     cpu
```

```
 -67960.1327      1.0000 17481  21:50:56    21.3
 -67960.1321      1.0000 17481  21:51:06    10.1

Finished on: Mon Jul 15 21:51:07 2013

LogLikelihood not converged

asreml 3.0-1 (24 October 2012), Library: 3.0hj (15 November 2011), X86_64
     LogLik         S2      DF      wall     cpu
 -67960.1302      1.0000 17481  21:51:30    21.7
 -67960.1300      1.0000 17481  21:51:40     9.3
 -67960.1295      1.0000 17481  21:51:49     9.9

Finished on: Mon Jul 15 21:51:50 2013

LogLikelihood not converged

asreml 3.0-1 (24 October 2012), Library: 3.0hj (15 November 2011), X86_64
     LogLik         S2      DF      wall     cpu
 -67960.1260      1.0000 17481  21:52:14    21.7
 -67960.1251      1.0000 17481  21:52:23     9.5
 -67960.1235      1.0000 17481  21:52:33     9.5
 -67960.1216      1.0000 17481  21:52:42     9.6

Finished on: Mon Jul 15 21:52:43 2013

LogLikelihood not converged

asreml 3.0-1 (24 October 2012), Library: 3.0hj (15 November 2011), X86_64
     LogLik         S2      DF      wall     cpu
 -67960.1207      1.0000 17481  21:53:06    21.7
 -67960.1202      1.0000 17481  21:53:16    10.0
 -67960.1193      1.0000 17481  21:53:26     9.6
 -67960.1181      1.0000 17481  21:53:37    10.9
 -67960.1176      1.0000 17481  21:53:48    10.7

Finished on: Mon Jul 15 21:53:48 2013

LogLikelihood not converged

asreml 3.0-1 (24 October 2012), Library: 3.0hj (15 November 2011), X86_64
     LogLik         S2      DF      wall     cpu
 -67960.1172      1.0000 17481  21:54:13    22.6
 -67960.1172      1.0000 17481  21:54:22     9.6
 -67960.1172      1.0000 17481  21:54:32     9.6
 -67960.1171      1.0000 17481  21:54:41     9.6

Finished on: Mon Jul 15 21:54:42 2013

LogLikelihood Converged
```

```
                                    Change(%)
fa(Expt, 2):ped(Tree)!Expt.FR203_3.var      1.71
fa(Expt, 2):ped(Tree)!Expt.FR353_2.fa2     12.97

> met.tree.asr2 <- temp.asr
> nicemet <- summary(met.tree.asr2,nice=T)$nice
> ped.lam.met <- matrix(nicemet[["fa(Expt, 2):ped(Tree)"]][-c(1:nemet)],nemet,2)
> ped.psi.met <- diag(nicemet[["fa(Expt, 2):ped(Tree)"]][1:nemet])
> Ga.met <- ped.lam.met%*%t(ped.lam.met) + ped.psi.met
> cov2cor(Ga.met)

          [,1]      [,2]      [,3]      [,4]      [,5]      [,6]
[1,] 1.0000000 0.5128809 0.5440170 0.6664543 0.6828490 0.2913169
[2,] 0.5128809 1.0000000 0.4561200 0.6141531 0.6951902 0.1370698
[3,] 0.5440170 0.4561200 1.0000000 0.9078956 0.7216932 0.8124323
[4,] 0.6664543 0.6141531 0.9078956 1.0000000 0.8909893 0.7033778
[5,] 0.6828490 0.6951902 0.7216932 0.8909893 1.0000000 0.3731633
[6,] 0.2913169 0.1370698 0.8124323 0.7033778 0.3731633 1.0000000

> ide.lam.met <- matrix(nicemet[["fa(TExpt):Tclone"]][-c(1:ne)],ne,1)
> ide.psi.met <- diag(nicemet[["fa(TExpt):Tclone"]][1:ne])
> Ge.met <- ide.lam.met%*%t(ide.lam.met) + ide.psi.met
> cov2cor(Ge.met)

          [,1]      [,2]      [,3]
[1,] 1.0000000 0.8401434 0.7317912
[2,] 0.8401434 1.0000000 0.6148095
[3,] 0.7317912 0.6148095 1.0000000
```

One last point to note, is that we have found that the updating algorithm used by the current version of `asreml()` is not often robust for fitting higher order FA variance models. To overcome this problem, we provide a *beta* version of a function which seems to work very well in practice, but has little theoretical justification. This is work in progress as we are developing a more robust and theoretically sound solution to this problem.

6.4.2 Example 5: ARA model

The final model we demonstrate is the ARA model for this MET data-set. We will demonstrate this using forward selections for all clones. The following presents the syntax to add the clones to the parental pedigree data-frame, and modify columns containing the female and male parents for each clone in the three trials

```
> ##########################
> # now for the ARA model
> # we first have to fix up the pedigree
> # to add forward selections to the parents file
> # and then change Fcln and Mcln
> ##########################
```

```
> met.df$Fcln.orig <- met.df$Fcln
> met.df$Mcln.orig <- met.df$Mcln
> temp.df <- subset(met.df,Check==1 & is.element(Expt,met.pclon))
> length(unique(temp.df$Tree)) # 542
> temp.df <- subset(temp.df,!duplicated(Tree))[,names(met.parent.df)]
> head(temp.df)
> temp.ped <- rbind(met.parent.df,temp.df[,names(met.parent.df)])
> nrow(temp.ped) # 844 = 302+542
> c(nrow(temp.ped),nrow(temp.ped[!duplicated(temp.ped$Tree),])) # [1] 844 844
> met.fsparent.df <- temp.ped
> ##############
> # now change the Fcln and Mcln in the data file to be themselves
> # select clone trials and Check==1
> # only
> ##############
> temp <- as.character(met.df$Fcln)
> temp[is.element(met.df$Expt,met.pclon) & met.df$Check==1] <-
+     as.character(met.df$Tree)[is.element(met.df$Expt,met.pclon) & met.df$Check==1]
> met.df$Fcln <- factor(temp)
> temp <- as.character(met.df$Mcln)
> temp[is.element(met.df$Expt,met.pclon) & met.df$Check==1] <-
+     as.character(met.df$Tree)[is.element(met.df$Expt,met.pclon) & met.df$Check==1]
> met.df$Mcln <- factor(temp)
> ###################
> # now the approximation ignoring inbreeding
> # with fsparent
> ####################
> temp <- asreml.Ainverse(met.fsparent.df)
> table(temp$inbreeding)
> met.fsparents.ainv <- temp$ginv
```

There are now a total of 844 clones and parents in these data.

We repeat the sequence of model fits and approaches which we used for the full tree model, starting with the diagonal variance model.

```
> met.df$zero <- rep(0,nrow(met.df))
> ######################3
> # now ready to to the ARA model for diag + diag
> #
> ######################
> met.aram.asr <- asreml(dbh~Expt,
+                       random=~str(~Expt:ped(Fcln):zero +
+                                   and(Expt:ped(Fcln),0.5) +
+                                   and(Expt:ped(Mcln),0.5),~diag(Expt):ped(Fcln)) +
+                       at(Expt):Replicate +
+                       at(Expt,met.pset):Replicate:Setgroup +
+                       at(Expt,met.pblk):Replicate:Iblk +
+                       diag(TExpt):Tclone,
```

```
+                     rcov=~at(Expt):units,data=met.df,
+                     ginverse=list(Fcln=met.fsparents.ainv,Mcln=met.fsparents.ainv),
+                     na.method.X='include', workspace=80e6,sparse=~Check)

asreml 3.0-1 (24 October 2012), Library: 3.0hj (15 November 2011), X86_64
     LogLik         S2      DF      wall     cpu
 -69144.6033      1.0000 17481  21:54:45     0.6 (1 restrained)
 -68673.6238      1.0000 17481  21:54:45     0.4
 -68262.6361      1.0000 17481  21:54:46     0.4
 -68086.9219      1.0000 17481  21:54:46     0.4
 -68080.6807      1.0000 17481  21:54:46     0.4
 -68080.6261      1.0000 17481  21:54:47     0.4
 -68080.6253      1.0000 17481  21:54:47     0.4
 -68080.6253      1.0000 17481  21:54:48     0.4

Finished on: Mon Jul 15 21:54:48 2013

LogLikelihood Converged
```

The computing time for this model is still very fast, even though we have more than doubled the number of parents by including forward selections.

Next we fit the FA(1) variance model to both terms, achieving convergence using default starting values

```
> ######################3
> # go straight to fa + fa
> # worked!!!
> #
> ######################
> met.aram.asr1 <- asreml(dbh~Expt,
+                     random=~str(~Expt:ped(Fcln):zero +
+                                 and(Expt:ped(Fcln),0.5) +
+                                 and(Expt:ped(Mcln),0.5),~fa(Expt):ped(Fcln)) +
+                     at(Expt):Replicate +
+                     at(Expt,met.pset):Replicate:Setgroup +
+                     at(Expt,met.pblk):Replicate:Iblk +
+                     fa(TExpt):Tclone,
+                     rcov=~at(Expt):units,data=met.df,
+                     ginverse=list(Fcln=met.fsparents.ainv,Mcln=met.fsparents.ainv),
+                     na.method.X='include', workspace=80e6,sparse=~Check)

asreml 3.0-1 (24 October 2012), Library: 3.0hj (15 November 2011), X86_64
     LogLik         S2      DF      wall     cpu
 -70273.4683      1.0000 17481  21:54:52     1.9 (7 restrained)
 -68872.3835      1.0000 17481  21:54:53     1.2 (12 restrained)
 -68448.3117      1.0000 17481  21:54:54     1.1 (11 restrained)
 -68215.2590      1.0000 17481  21:54:55     1.6 (10 restrained)
 -68107.9507      1.0000 17481  21:54:56     1.0 (10 restrained)
```

```
 -68048.7227      1.0000 17481  21:54:57     1.0 (10 restrained)
 -68020.1468      1.0000 17481  21:54:58     1.0 (10 restrained)
 -67997.2179      1.0000 17481  21:54:59     1.0 (1 restrained)
 -67973.6431      1.0000 17481  21:55:00     1.0 (1 restrained)
 -67967.8871      1.0000 17481  21:55:01     1.0 (1 restrained)
 -67965.6369      1.0000 17481  21:55:03     1.6
 -67963.7020      1.0000 17481  21:55:04     1.0
 -67963.3900      1.0000 17481  21:55:05     0.9

Finished on: Mon Jul 15 21:55:05 2013

LogLikelihood not converged

> met.aram.asr1 <- update(met.aram.asr1)

asreml 3.0-1 (24 October 2012), Library: 3.0hj (15 November 2011), X86_64
     LogLik          S2      DF      wall     cpu
 -67963.3739      1.0000 17481  21:55:09     1.7
 -67963.3738      1.0000 17481  21:55:10     0.9
 -67963.3735      1.0000 17481  21:55:11     1.0
 -67963.3731      1.0000 17481  21:55:11     1.0
 -67963.3729      1.0000 17481  21:55:12     1.0

Finished on: Mon Jul 15 21:55:13 2013

LogLikelihood Converged
                                    Change(%)
fa(TExpt):Tclone!TExpt.FR353_2.var      1.27

> ssmetaram <- summary(met.aram.asr1)$varcomp
```

We conclude by fitting the FA(2) variance model to the additive genetic by environment effects below:

```
> ########################3
> # go for fa2 + fa
> # worked!!!
> #
> ########################
> met.aram.sv <- asreml(dbh~Expt,
+                       random=~str(~Expt:ped(Fcln):zero +
+                                   and(Expt:ped(Fcln),0.5) +
+                                   and(Expt:ped(Mcln),0.5),~fa(Expt,2):ped(Fcln)) +
+                       at(Expt):Replicate +
+                       at(Expt,met.pset):Replicate:Setgroup +
+                       at(Expt,met.pblk):Replicate:Iblk +
+                       fa(TExpt):Tclone,start.values=T,
+                       rcov=~at(Expt):units,data=met.df,
+                       ginverse=list(Fcln=met.fsparents.ainv,Mcln=met.fsparents.ainv),
```

```
+                     na.method.X='include', workspace=80e6,sparse=~Check)
> temp <- met.aram.sv$gammas.table
> temp[grep('.*Fcln',temp$Gamma,invert=T),'Value'] <-
+     ssmetaram[grep('.*Fcln',dimnames(ssmetaram)[[1]],invert=T),'component']
> fa2.svs <- c(summary(met.aram.asr1,nice=T)$nice[["Expt:ped(Fcln):zero"]],
+              c(0,rep(.1,nemet-1)))
> temp[grep('.*Fcln',temp$Gamma,invert=F),'Value'] <- fa2.svs
> met.aram.asr2 <- asreml(dbh~Expt,
+                     random=~str(~Expt:ped(Fcln):zero +
+                                 and(Expt:ped(Fcln),0.5) +
+                                 and(Expt:ped(Mcln),0.5),~fa(Expt,2):ped(Fcln)) +
+                     at(Expt):Replicate +
+                     at(Expt,met.pset):Replicate:Setgroup +
+                     at(Expt,met.pblk):Replicate:Iblk +
+                     fa(TExpt):Tclone,R.param=temp,G.param=temp,
+                     rcov=~at(Expt):units,data=met.df,maxit=10,
+                     ginverse=list(Fcln=met.fsparents.ainv,Mcln=met.fsparents.ainv),
+                     na.method.X='include', workspace=80e6,sparse=~Check)

asreml 3.0-1 (24 October 2012), Library: 3.0hj (15 November 2011), X86_64
     LogLik         S2      DF      wall     cpu
 -67963.3731      1.0000 17481  21:55:19     2.5
Logliklihood decreased to  -68384.93 - trying again with reduced updates
 -67967.0849      1.0000 17481  21:55:22     3.6
 -67966.2071      1.0000 17481  21:55:24     1.8
 -67963.9381      1.0000 17481  21:55:26     1.8
 -67964.9349      1.0000 17481  21:55:28     1.8 (1 restrained)
 -67963.2956      1.0000 17481  21:55:29     1.8 (1 restrained)
 -67960.7400      1.0000 17481  21:55:31     1.8 (1 restrained)
 -67960.1194      1.0000 17481  21:55:33     1.7 (1 restrained)
 -67960.0062      1.0000 17481  21:55:35     2.3
 -67959.9654      1.0000 17481  21:55:37     1.6

Finished on: Mon Jul 15 21:55:37 2013

LogLikelihood not converged

> temp.asr  <- update.fa(met.aram.asr2,startsearch=F,endsearch=T,midsearch=F,
+                        endloops=5,enditer=3,convtest=F)

asreml 3.0-1 (24 October 2012), Library: 3.0hj (15 November 2011), X86_64
     LogLik         S2      DF      wall     cpu
 -67959.9596      1.0000 17481  21:55:41     2.3
 -67959.9591      1.0000 17481  21:55:43     1.6
 -67959.9580      1.0000 17481  21:55:45     1.7

Finished on: Mon Jul 15 21:55:45 2013

LogLikelihood not converged
```

```
asreml 3.0-1 (24 October 2012), Library: 3.0hj (15 November 2011), X86_64
     LogLik         S2      DF      wall     cpu
 -67959.9553      1.0000 17481  21:55:49     2.3
 -67959.9539      1.0000 17481  21:55:51     1.6
 -67959.9498      1.0000 17481  21:55:52     1.7

Finished on: Mon Jul 15 21:55:53 2013

LogLikelihood not converged

asreml 3.0-1 (24 October 2012), Library: 3.0hj (15 November 2011), X86_64
     LogLik         S2      DF      wall     cpu
 -67959.9483      1.0000 17481  21:55:57     2.3
 -67959.9481      1.0000 17481  21:55:59     1.7
 -67959.9479      1.0000 17481  21:56:00     1.6

Finished on: Mon Jul 15 21:56:00 2013

LogLikelihood not converged

asreml 3.0-1 (24 October 2012), Library: 3.0hj (15 November 2011), X86_64
     LogLik         S2      DF      wall     cpu
 -67959.9469      1.0000 17481  21:56:05     2.3
 -67959.9466      1.0000 17481  21:56:06     1.6
 -67959.9461      1.0000 17481  21:56:08     1.7

Finished on: Mon Jul 15 21:56:08 2013

LogLikelihood not converged

asreml 3.0-1 (24 October 2012), Library: 3.0hj (15 November 2011), X86_64
     LogLik         S2      DF      wall     cpu
 -67959.9457      1.0000 17481  21:56:13     2.4
 -67959.9456      1.0000 17481  21:56:14     1.7
 -67959.9455      1.0000 17481  21:56:16     1.6

Finished on: Mon Jul 15 21:56:16 2013

LogLikelihood not converged

> met.aram.asr2 <- temp.asr
> temp.asr  <- update.fa(met.aram.asr2,startsearch=F,endsearch=T,midsearch=F,
+                        endloops=5,enditer=3,convtest=F)

asreml 3.0-1 (24 October 2012), Library: 3.0hj (15 November 2011), X86_64
     LogLik         S2      DF      wall     cpu
 -67959.9452      1.0000 17481  21:56:20     2.3
 -67959.9451      1.0000 17481  21:56:22     1.6
 -67959.9449      1.0000 17481  21:56:24     1.7
```

```
Finished on: Mon Jul 15 21:56:24 2013

LogLikelihood not converged

asreml 3.0-1 (24 October 2012), Library: 3.0hj (15 November 2011), X86_64
     LogLik         S2      DF      wall     cpu
 -67959.9447      1.0000 17481  21:56:28     2.3
 -67959.9447      1.0000 17481  21:56:30     1.6
 -67959.9447      1.0000 17481  21:56:31     1.6

Finished on: Mon Jul 15 21:56:31 2013

LogLikelihood not converged

asreml 3.0-1 (24 October 2012), Library: 3.0hj (15 November 2011), X86_64
     LogLik         S2      DF      wall     cpu
 -67959.9446      1.0000 17481  21:56:36     2.3
 -67959.9445      1.0000 17481  21:56:37     1.6
 -67959.9444      1.0000 17481  21:56:39     1.6

Finished on: Mon Jul 15 21:56:39 2013

LogLikelihood not converged

asreml 3.0-1 (24 October 2012), Library: 3.0hj (15 November 2011), X86_64
     LogLik         S2      DF      wall     cpu
 -67959.9444      1.0000 17481  21:56:44     2.3
 -67959.9444      1.0000 17481  21:56:45     1.6
 -67959.9444      1.0000 17481  21:56:47     1.7

Finished on: Mon Jul 15 21:56:47 2013

LogLikelihood not converged

asreml 3.0-1 (24 October 2012), Library: 3.0hj (15 November 2011), X86_64
     LogLik         S2      DF      wall     cpu
 -67959.9444      1.0000 17481  21:56:51     2.3
 -67959.9443      1.0000 17481  21:56:53     1.7
 -67959.9443      1.0000 17481  21:56:55     1.7

Finished on: Mon Jul 15 21:56:55 2013

LogLikelihood not converged

> met.aram.asr2 <- temp.asr
> temp.asr  <- update.fa(met.aram.asr2,startsearch=F,endsearch=T,midsearch=F,
+                        endloops=5,enditer=3,convtest=F)
```

```
asreml 3.0-1 (24 October 2012), Library: 3.0hj (15 November 2011), X86_64
     LogLik         S2      DF      wall     cpu
 -67959.9443      1.0000 17481  21:56:59     2.3
 -67959.9443      1.0000 17481  21:57:01     1.7
 -67959.9443      1.0000 17481  21:57:03     1.7

Finished on: Mon Jul 15 21:57:03 2013

LogLikelihood not converged

asreml 3.0-1 (24 October 2012), Library: 3.0hj (15 November 2011), X86_64
     LogLik         S2      DF      wall     cpu
 -67959.9443      1.0000 17481  21:57:07     2.3
 -67959.9443      1.0000 17481  21:57:09     1.8
 -67959.9442      1.0000 17481  21:57:11     1.6

Finished on: Mon Jul 15 21:57:11 2013

LogLikelihood not converged

asreml 3.0-1 (24 October 2012), Library: 3.0hj (15 November 2011), X86_64
     LogLik         S2      DF      wall     cpu
 -67959.9442      1.0000 17481  21:57:15     2.3
 -67959.9442      1.0000 17481  21:57:17     1.6
 -67959.9442      1.0000 17481  21:57:19     1.7

Finished on: Mon Jul 15 21:57:19 2013

LogLikelihood not converged

asreml 3.0-1 (24 October 2012), Library: 3.0hj (15 November 2011), X86_64
     LogLik         S2      DF      wall     cpu
 -67959.9442      1.0000 17481  21:57:23     2.3
 -67959.9442      1.0000 17481  21:57:25     1.7
 -67959.9442      1.0000 17481  21:57:26     1.6

Finished on: Mon Jul 15 21:57:26 2013

LogLikelihood not converged

asreml 3.0-1 (24 October 2012), Library: 3.0hj (15 November 2011), X86_64
     LogLik         S2      DF      wall     cpu
 -67959.9442      1.0000 17481  21:57:31     2.3
 -67959.9442      1.0000 17481  21:57:32     1.6
 -67959.9442      1.0000 17481  21:57:34     1.7

Finished on: Mon Jul 15 21:57:34 2013

LogLikelihood not converged
```

```
> met.aram.asr2 <- temp.asr
> temp.asr  <- update.fa(met.aram.asr2,startsearch=F,endsearch=T,midsearch=F,
+                        endloops=5,enditer=4,convtest=F)

asreml 3.0-1 (24 October 2012), Library: 3.0hj (15 November 2011), X86_64
     LogLik          S2      DF      wall     cpu
 -67959.9442      1.0000 17481  21:57:39     2.3
 -67959.9442      1.0000 17481  21:57:40     1.6
 -67959.9442      1.0000 17481  21:57:42     1.7
 -67959.9442      1.0000 17481  21:57:43     1.6

Finished on: Mon Jul 15 21:57:44 2013

LogLikelihood Converged
                                       Change(%)
Expt:ped(Fcln):zero!Expt.FR353_2.fa2      1.52

> met.aram.asr2 <- temp.asr
> nicemet <- summary(met.aram.asr2,nice=T)$nice
> ped.lam.met <- matrix(nicemet[["Expt:ped(Fcln):zero"]][-c(1:nemet)],nemet,2)
> ped.psi.met <- diag(nicemet[["Expt:ped(Fcln):zero"]][1:nemet])
> Ga.met <- ped.lam.met%*%t(ped.lam.met) + ped.psi.met
> cov2cor(Ga.met)

          [,1]      [,2]      [,3]      [,4]      [,5]      [,6]
[1,] 1.0000000 0.5132772 0.5438942 0.6636169 0.6821364 0.2843062
[2,] 0.5132772 1.0000000 0.4557339 0.6108205 0.6941686 0.1289090
[3,] 0.5438942 0.4557339 1.0000000 0.9122967 0.7273644 0.8121810
[4,] 0.6636169 0.6108205 0.9122967 1.0000000 0.8909773 0.7026377
[5,] 0.6821364 0.6941686 0.7273644 0.8909773 1.0000000 0.3732160
[6,] 0.2843062 0.1289090 0.8121810 0.7026377 0.3732160 1.0000000

> ide.lam.met <- matrix(nicemet[["fa(TExpt):Tclone"]][-c(1:ne)],ne,1)
> ide.psi.met <- diag(nicemet[["fa(TExpt):Tclone"]][1:ne])
> Ge.met <- ide.lam.met%*%t(ide.lam.met) + ide.psi.met
> cov2cor(Ge.met)

          [,1]      [,2]      [,3]
[1,] 1.0000000 0.8395701 0.7322098
[2,] 0.8395701 1.0000000 0.6147415
[3,] 0.7322098 0.6147415 1.0000000
```

Bibliography

Apiolaza, L. (2012). Basic density of radiata pine in new zealand: genetic and environmental factors. *Tree Genetics and Genomes* **8**, 87–9.

Baltunis, B., Gapare, W., & Wu, H. (2010). Genetic parameters and genotype by environment interaction in radiata pine for growth and wood quality traits in australia. *Silvae Genetica* **59**, 2–3.

Beeck, C., Cowling, W., Smith, A., & Cullis, B. (2010). Analysis of yield and oil from a series of canola breeding trials. Part I: Fitting factor analytic models with pedigree information. *Genome* **53**, 992–1001.

Brien, C. & Demetrio, C. (2009). Formulating mixed models for experiments, including longitudinal experiments. *Journal of Agricultural, Biological and Environmental Statistics* **14**, 253–280.

Butler, D. G., Cullis, B. R., Gilmour, A. R., & Gogel, B. J. (2009a). ASReml-R reference manual, release 3. Technical report, Queensland Department of Primary Industries.

Butler, D. G., Cullis, B. R., Gilmour, A. R., & Gogel, B. J. (2009b). *ASReml-R Reference Manual, Release 3*. Queensland Department of Primary Industries.

Cullis, B., Smith, A., Beeck, C., & Cowling, W. (2010). Analysis of yield and oil from a series of canola breeding trials. Part II: Exploring VxE using factor analysis. *Genome* **53**, 1002–1016.

Gilmour, A. R., Cullis, B. R., Welham, S. J., Gogel, B. J., & Thompson, R. (2009). *ASREML, Reference Manual - Release 3*. VSN International.

Gilmour, A. R., Thompson, R., & Cullis, B. R. (1995). AI, an efficient algorithm for REML estimation in linear mixed models. *Biometrics* **51**, 1440–1450.

Kelly, A., Smith, A., Eccleston, J., & Cullis, B. (2007). The accuracy of varietal selection using factor analytic models for multi-environment plant breeding trials. *Crop Science* **47**, 1063–1070.

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

Lane, P. W. & Nelder, J. A. (1982). Analysis of covariance and standardisation as instances of prediction. *Biometrics* **82**, 613–621.

Lynch, M. & Walsh, B. (1998). *Genetics and analysis of quantitative traits*. Sinauer Associates.

Martin, R. J. (1979). A subclass of lattice processes applied to a problem in planar sampling. *Biometrika* **66**, 209–217.

McCullagh, P. & Nelder, J. A. (1994). *Generalized Linear Models*. Chapman and Hall, London, 2 edition.

Meuwissen, T. & Luo, Z. (1992). Computing inbreeding coefficients in large populations. *Genetics, Selection and Evolution* **24**, 305–315.

Mrode, R. (1995). *Linear models for the prediction of animal breeding values*. CABI Publishing.

Nelder, J. A. (1994). The statistics of linear models: back to basics. *Statistics and Computing* **4**, 221–234.

Patterson, H. D. & Thompson, R. (1971). Recovery of interblock information when block sizes are unequal. *Biometrika* **31**, 545–554.

Quass, R. L. & Pollack, E. (1980). Mixed model methodology for farm and ranch beef cattle testing programs. *Journal of Animal Science* **51**, 1277–1287.

Raymond, C. (2011). Genotype by environment interactions for pinus radiata in new south wales, australia. *Tree Genetics & Genomes* **7**, 819–833.

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 and sons, inc.

Smith, A., Cullis, B., & Thompson, R. (2005). The analysis of crop cultivar breeding and evaluation trials: an overview of current mixed model approaches. *Journal of Agricultural Science, Cambridge* **143**, 449–462.

Smith, A., Cullis, B. R., & Thompson, R. (2001). Analyzing variety by environment data using multiplicative mixed models and adjustments for spatial field trend. *Biometrics* **57**, 1138–1147.

Stefanova, K., Smith, A., & Cullis, B. (2009). Enhanced diagnostics for the spatial analysis of field trials. *Journal of Agricultural, Biological and Environmental Statistics* **14**, 1–19.

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

Thompson, R., Cullis, B., Smith, A., & Gilmour, A. (2003). A sparse implementation of the average information algorithm for factor analytic and reduced rank variance models. *Australian and New Zealand Journal of Statistics* **45**, 445–459.

Welham, S. J., Cullis, B. R., Gogel, B. J., Gilmour, A. R., & Thompson, R. (2004). Prediction in mixed linear models. *Australian and New Zealand Journal of Statistics* **46**, 325–347.

White, I., Rainer, R. Knap, P., & Brotherstone, S. (2006). Variance components for survival of piglets at farrowing using a reduced animal model. *Genetic Selection and Evolution* **38**, 359–370.

Wilkinson, G. N. & Rogers, C. E. (1973). Symbolic description of factorial models for analysis of variance. *Applied Statistics* **22**, 392–399.

Yates, F. (1935). Complex experiments. *Journal of the Royal Statistical Society B* **2**, 181–247.