ASReml Update

What's new in Release 4.1

B J Gogel[1]
A R Gilmour[4]
S J Welham[4]
B R Cullis[2,3]
R Thompson[5]

[1]The University of Adelaide, Waite Campus, PMB 1, Glen Osmond SA, 5064, Australia
[2]School of Mathematics and Statistics, University of Wollongong, Wollongong, NSW, 2550, Australia
[3]Computational Informatics, CSIRO Canberra ACT, Australia
[4]VSN International
[5]Centre for Mathematical and Computational Biology, and Department of Biomathematics and Bioinformatics, Rothamsted Research, Harpenden AL5 2JQ, United Kingdom

December 20, 2014

ASReml Update: What's new in Release 4.1

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

Published by:

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

Copyright Notice

The correct bibliographical reference for this document is:

Gogel, B.G., Gilmour, A.R., Cullis, B.R., Welham, S.J. and Thompson, R. 2014. ASReml Update: What's new in Release 4.1. VSN International Ltd, Hemel Hempstead, HP1 1ES, UK.

Author email addresses

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

Arthur Gilmour acknowledges the grace of God through Jesus Christ who enables this research to proceed. *Be exalted O God, above the heavens: and Thy glory above all the earth.* Psalm 108:5.

Preface

Developments in ASReml Release 4

There are a number of new developments in Release 4. These are described in Chapter 1. Developments associated with input include generating initial values, generating a template to allow an alternative way of presenting parametric information associated with variance structures, new facilities for reading in data files and defining factor names and improved facilities for reading relationship matrices. Among the developments associated with analysis are making it easier to specify functions of variance parameters using names rather than numbers, fitting factor effects with large random regression models, such as commonly used with marker data, fitting linear relationships among variance structure parameters and calculating information criteria. The developments associated with output include writing out design matrices.

New model specification using a functional approach

The main development in ASReml Release 4 is a new model specification using a *functional approach*. Prior to Release 4 a *structural* specification was used in which variance models were applied by imposing variance structures on random model terms and/or the residual error term after the mixed model had been specified. In this case, the variance models were presented in a separate part of the input file. The functional specification offers a simpler alternative to the structural specification in which the variance structures for random model terms and the residual error term are specified in the linear mixed model definition by wrapping terms with the required variance model function. This functional approach is more concise, less error-prone and more automatic for specifying multi-section residual variances.

The functional model specification has been developed from the model implementation of ASReml-R and offers a more seamless movement between platforms. A complete discussion of the functional specification, with reference to the structural specification, is presented in Chapter 2. Chapter 3 takes examples from the User Guide and compares the functional and structural specifications.

Who should read this document?

This document is for existing users who wish to keep abreast of the developments with Release 4. Chapter 2 is primarily for existing users who wish to migrate to the functional model specification. Users who are new to ASReml with Release 4 are referred to the ASReml User Guide for this version.

Contents

List of Tables

1 Developments in ASReml Release 4

This chapter is a summary of the developments in Release 4 associated with

- **input** (Section 1.1) including the generation of initial values, the generation of a template to allow an alternative way of presenting parametric information associated with variance structures and improved facilities for reading relationship matrices

- **analysis** (Section 1.2) including making it easier to specify functions of variance parameters using names rather than numbers, fitting factor effects with large random regression models,such as commonly used with marker data, fitting linear relationships among variance structure parameters and calculating information criteria

- **output** (Section 1.3) including writing out design matrices.

Another development is the implementation of an alternative way of specifying the mixed model. The distinguishing feature of this specification is that the variance models for individual terms are specified by applying variance model functions to the terms in the model definition line(s). This is in contrast to the structural specification of versions prior to Release 4 in which the variance models were only imposed after the mixed model definition line(s). Some users might find this functional specification more natural and less error prone. There is a comprehensive discussion of the specification in Chapter 2 and Chapter 3 compares the structural and functional specifications for a range of User Guide examples.

1.1 Improvements to input

1.1.1 Generating initial values for variance structure parameters

In ASReml 3 it was necessary to supply initial values for variance structure parameters except for the default variance structure for a random model term, where the default initial variance (ratio) parameter value was 0.1. In ASReml 4, it is not generally necessary to supply initial values. In this release, if the user uses * instead of the initial value(s) then ASReml provides initial values for variance structure parameters based on knowledge of the phenotypic variance of the response. Occasionally these initial values are not adequate and more appropriate values will need to be supplied by the user, perhaps using prior information

or results from a simplified analysis.

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

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

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

generates a `.tsv` file.

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

5, "units.us(Trait);us(Trait)_1", G, P, 4.7911110 , 5, 1
6, "units.us(Trait);us(Trait)_2", G, P, 5.0231481 , 6, 1
7, "units.us(Trait);us(Trait)_3", G, P, 15.298889 , 7, 1
8, "units.us(Trait);us(Trait)_4", G, P, 4.8438271 , 8, 1
9, "units.us(Trait);us(Trait)_5", G, P, 11.264815 , 9, 1
10, "units.us(Trait);us(Trait)_6", G, P, 26.095692 , 10, 1
11, "units.us(Trait);us(Trait)_7", G, P, 4.6882715 , 11, 1
12, "units.us(Trait);us(Trait)_8", G, P, 10.824074 , 12, 1
13, "units.us(Trait);us(Trait)_9", G, P, 27.332887 , 13, 1
14, "units.us(Trait);us(Trait)_10", G, P, 71.875403 , 14, 1
15, "units.us(Trait);us(Trait)_11", G, P, 3.9083333 , 15, 1
16, "units.us(Trait);us(Trait)_12", G, P, 10.292592 , 16, 1
```

1.1 Improvements to input

```
17, "units.us(Trait);us(Trait)_13", G, P, 34.137962 , 17, 1
18, "units.us(Trait);us(Trait)_14", G, P, 69.287036 , 18, 1
19, "units.us(Trait);us(Trait)_15", G, P, 141.97296 , 19, 1
```

Parameter constraints and initial values can be changed by editing the values in the `PSpace` and `Initial_value` columns. Scale relationships can be introduced by noting that the full set of parameters can be related to a subset of parameters and scale factors such as

*parameter = subset parameter * scale*

or

`GN` *column parameter*, `RP_GN` *column parameter* * `RP_scale` *value*

where `GN`, `RP_GN` and `RP_scale` are columns in the `.tsv` file. The relationships generated by

```
VCC 2
5 6 8 11 15 7 * 2 9 * 2 12 * 2 16 * 2 #parameters 6 8 11 15 are equal to 5
#7 9 12 16 are twice 5
10 13 17 #parameters 13 and 17 are equal to 10
#the full set of parameters 5-19 can therefore be expressed in terms of the subset parameters
5, 10 ,14, 18 and 19
```

can be introduced by editing the `RN_GN` and `RP_scale` columns. Some users would prefer to insert initial values into this `.tsv` file under the `Initial_value` column. As an example, the file below contains values based on using 4.8, 26, 70, 35 and 70 for parameters 5, 10, 14, 18 and 19. The data values in the `.tsv` file become

```
# GN, Term, Type, PSpace, Initial_value, RP_GN, RP_scale.
5, "units.us(Trait);us(Trait)_1", G, P, 4.8 , 5, 1.0000
6, "units.us(Trait);us(Trait)_2", G, P, 4.8 , 5, 1.0000
7, "units.us(Trait);us(Trait)_3", G, P, 9.6 , 5, 2.0000
8, "units.us(Trait);us(Trait)_4", G, P, 4.8 , 5, 1.0000
9, "units.us(Trait);us(Trait)_5", G, P, 9.6 , 5, 2.0000
10, "units.us(Trait);us(Trait)_6", G, P, 26 ,10, 1.0000
11, "units.us(Trait);us(Trait)_7", G, P, 4.8 , 5, 1.0000
12, "units.us(Trait);us(Trait)_8", G, P, 9.6 , 5, 2.0000
13, "units.us(Trait);us(Trait)_9", G, P, 26 , 10, 1.0000
14, "units.us(Trait);us(Trait)_10", G, P, 70 , 14, 1.0000
15, "units.us(Trait);us(Trait)_11", G, P, 4.8 , 5, 1.0000
16, "units.us(Trait);us(Trait)_12", G, P, 9.6 , 5, 2.0000
17, "units.us(Trait);us(Trait)_13", G, P, 26 , 10, 1.0000
18, "units.us(Trait);us(Trait)_14", G, P, 35 , 18, 1.0000
19, "units.us(Trait);us(Trait)_15", G, P, 70 , 19, 1.0000
```

Sometimes users wish to rerun a job making changes to the final values, parametric constraints and relationships (equality and scale) between parameters. A file `.msv` is produced, similar to `.tsv` but containing final values that can be edited and used with `!MSV`. If `!TSV`

(or !MSV) is specified ASReml will read the current (created with the same PART number) .tsv (or .msv) file. If there is no current .tsv (or .msv file), a non-current (produced from a different PART of the same job) .tsv (or .msv) file will be read.

Alternative ways of specifying !TSV and !MSV are !CONTINUE 2 and !CONTINUE 3 and these qualifiers can be used as options on the command line as -C2 and -C3. Note that the constraints in the .tsv/.msv files take precedence over those in the .as file.

If f=filename, with no extension, is used with !CONTINUE, !TSV or !MSV, ASReml will use the file f.rsv, f.tsv or f.msv. If f=filename.xsv with x=r, t or m is used with !CONTINUE, !TSV or !MSV, ASReml will use the file f.xsv. If the specified file is not present, ASReml reverts to reading the previous .rsv file.

1.1.3 Skipping fields during data input

The meaning of the !SKIP n qualifier differs with context. It is most commonly specified after the name of a pedigree or data file to instruct ASReml to ignore the first n lines of the file (presumably heading lines). !SKIP can also be used to skip columns when reading in a data file. For example,

```
A !SKIP 1 B
```

instructs ASReml to skip 1 data field before reading the field to be labelled A.

A new qualifier, !CSKIP c, is now available to skip data fields (columns) when reading the data. It is an alternative to !SKIP n but is more logically placed. For example,

```
!CSKIP 1 A B
```

does the same as the above !SKIP command, that is, skips the first data field and reads the second and third data fields into variables A and B. Likewise,

```
!CSKIP n A B
```

skips the first n data fields and reads the next two fields into variables A and B.

We suggest using !CSKIP in preference to !SKIP for skipping data fields.

1.1.4 Setting the order of alphanumeric factor levels

An !L qualifier associated with the !A qualifier was available in Release 3 and pre-declared the levels of alphanumeric factors and their order. ASReml will now accept a filename argument where the first field of each line of the file contains the names of the levels. If the filename has Identity as the first name on the first line, the first line is ignored. The # character is treated as the beginning of a comment; the # itself and the following characters are ignored, as are blank lines. Consequently, a level label may not contain the character #. If the filename includes embedded blanks, or has no file extension, it must be enclosed in quotes:

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

When actually reading the data, any label that is not already defined is appended to the list when it is encountered. One use of this qualifier is to ensure that the data is coded according to the order of the identities (ID's) used when forming `.grm` or `.giv` files; supplying the order in a text file. For example, if the file `Idnames.txt` contains the list and has 1 header line, define `ID` as

```
ID !A 450 !L Idnames.txt !SKIP 1
```

where 450 is the number of levels. Note that in this context of defining a label file, ASReml interprets `!SKIP` as skipping a line rather than a column (field) in the names file. Refer to the User Guide for information on data file types.

1.1.5 Extra facilities for reading relationship matrix (including GRM) files

New model names

In previous releases of ASReml, a pedigree file could be associated with a factor, and a numerator relationship matrix was generated and could be used as a variance structure. Slightly illogically, this structure was specified in models using `AINV` (Additive Inverse Matrix). Similarly, generalised relationship matrices (`grm`) or their inverses (`giv`) could be read in and `giv()` and `GIV` could be used in the model specification. In Release 4, ASReml allows the use of `NRM` as a synonym for `AINV` and the use of `nrm(`*f*`)` on the model line. Furthermore, `GRM`*n* and `grm`*n*`(`*f*`)` can be used as synonyms for `GIV`*n* and `giv(`*f*`,`*n*`)` respectively, where *f* is the model term variable to which the structure is applied and *n* is the ordinal number of the `GIV`/`GRM` matrix being associated with *f*.

GRM files

GRM and GIV files will be read assuming single/double precision lower triangle row-wise binary format if the file extension is `.sgiv`, `.dgiv`, `.sgrm` or `.dgrm`.

`!PRECISION n` changes the value used to declare a singularity when inverting a GRM file from `1D-7` to `1D-n`.

`!SAVEGIV` on the GRM line has been extended to

`!SAVEGIV 3` which writes the GIV matrix as an `.sgiv` file, or

`!SAVEGIV 4` which writes the GIV matrix as a `.dgiv` file, where `.sgiv` is single precision lower triangle row-wise binary and `.dgiv` is double precision lower triangle row-wise binary.

Diagnostics when processing GRM files have been improved. ASReml now tests the GRM matrix and reports if it is not positive definite. In this case, 3 qualifiers may be used to

allow ASReml to proceed. If the matrix has positive and negative eigenvalues, !ND instructs ASReml to ignore the condition and proceed anyway. If the matrix is positive semi-definite (positive and zero eigenvalues), !PSD allows ASReml to introduce Lagrangian multipliers to accommodate linear dependencies and rows with zero elements, and allows ASReml to proceed. Linear dependencies occur, for example, when the list of individuals includes clones. Rows with zero elements occur when the GRM represents a dominance matrix, and the list of individuals includes fully inbred individuals which, by definition, have zero dominance variance. If the matrix has positive, zero and negative eigenvalues, !NSD may be used to allow ASReml to continue. The zero eigenvalues are handled as for !PSD.

Another example of !L (Section 1.1.4) is in analysis on data with 2 relationship matrices based on two separate pedigrees. ASReml only allows one pedigree file to be specified but can create an inverse relationship matrix and store the result in a GIV file. So, 2 relationship matrices based on two separate pedigrees may be used by generating a GIV file from one pedigree and then using that GIV file and the other pedigree in a subsequent run. To process the GIV file properly, we must also generate a file with identities as required for the GIV matrix. An example of this is if the file Hybrid.as includes

```
!PART 1
Mline !P
Fline !A
...
Mline.ped !GIV !DIAG #!GIV generates the file Hybrid1A.giv and !DIAG
#generates Hybrid1.aif which contains the identifier names
!PART 2 #reads in inverse additive relationship matrix generated in !PART 1
Mline !A !L Hybrid1.aif !SKIP 1#associates identifier names with levels of Mline
#used in giv file
Fline !P
...
Fline.ped !GIV !DIAG
Hybrid1_A.giv #formed in part 1 from Mline.ped
Hybrid.asd !SKIP 1
...
...  grm1(Mline) nrm(Fline) #using new synonyms and functions
```

1.1.6 Handling the # character in data files

The general (data line) qualifier !SPECIALCHAR causes # to be treated as an ordinary character in the data file, rather than to have its usual meaning, that is, to *ignore the rest of the input line*. ASReml generates a warning message when the # character appears in the data file and !SPECIALCHAR has not been used.

1.1.7 Extension to the !FILTER qualifier

The !FILTER f qualifier is used with the !SELECT s qualifier to select only the data records that have the value s in field f. This has been extended by addition of the qualifier !EXCLUDE e to ignore all data records that do have the value e in field f.

1.1.8 Using binary MBF files

The !MBF statement extracts covariate values corresponding to levels of a factor from an auxiliary file. This file may now be a binary format file, with file extension .bin indicating 32bit real binary numbers and .dbl indicating 64bit real binary values. Files with these formats can be easily created in a preliminary run using the !SAVE qualifier. The advantage of using a binary file is that reading the file is much quicker. This is important if the file has many fields and is being accessed repeatedly, for example

```
!CYCLE 1:1000
!MBF mbf(Geno) markers.dbl !key 1 !RFIELD $I !rename M$I
...  !r M$I
```

1.2 Improvements to analysis

1.2.1 Fitting linear relationships among variance structure parameters

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

VCM *first [intermediate] last new filename*

where *first* is the parameter number of the first parameter in the set, *intermediate* is required unless all the intermediate parameters are involved, *last* is the parameter number of the last parameter in the set, *new* is the number of new parameters
filename is the name of the file containing the design matrix.

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

```
Wolfinger Rat data
 treat !A
```

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

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

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

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

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

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

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

Note that if the user fits another model with differing numbers of variance structure parameters so that the variance structure parameters are renumbered, then all the user needs to

do to continue with the same relationships is to change the first and last parameters on the `VCM` line.

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

1.2.2 Data order, filling in the spatial grid

In Release 3, the qualifiers `!SECTION`, `!ROWFACTOR` and `!COLUMNFACTOR` in combination with an appropriate *Variance header line*, generated an AR1×AR1 variance structure for each section. That is, it created the internal structures to fit this model and wrote the lines which specified this structure to the `.res` so that they could be pasted into a revised `.as` file.

However, to fit this model, it is necessary that the complete grid of plots be represented in the data file, even though some of them may be missing a yield response. In ASReml 4, if these qualifiers are specified and the model term `mv` is fitted, ASReml will sort the data on the three fields specified and insert 'missing value' records to complete the row/column grid for each section.

1.2.3 Information criteria

For comparing nested models we recommend the REML likelihood ratio test (REMLRT), see Section 2.5 of the User Guide. The *Akaike Information Criterion* (AIC) and the *Bayesian Information Criterion* (BIC) are defined in equation 2.15 of the User Guide. The AIC and BIC are provided for the convenience of users but without any formal recommendation for their use. The number of parameters includes the number of linear parameters estimated and the number of variance structure parameters estimated, excluding variance parameters fixed at a boundary during the estimation procedure. The value used in calculating AIC and BIC is reported, giving the opportunity for the user to verify/modify this number.

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

1.2.4 Conditional coding

The facilities for manipulating job coding have been extended with the high level qualifiers in Table 1.1. These are high level in the sense that they can appear at any point in the command file, provided the result can be formed at that point.

1.2.5 Processing random terms as DENSE

ASReml uses link list matrix methods for the sparse equations and has always included random model terms in these sparse equations. However, in the case of GRM matrices, they are typically quite large (several thousand) and dense, and it would be more efficient

1.2 Improvements to analysis

generally to process them as such. Release 4 has a !GDENSE qualifier (set just before the model line). If !GDENSE is set and the first random term is a GRM term, its equations will be processed as DENSE. For an example with 3226 rows in the GRM matrix, !GDENSE reduced the iteration time from 196 to 175 seconds.

Table 1.1: High level qualifiers

qualifier	action
!ASSIGN *list*	An !ASSIGN *string* qualifier has been added to extend coding options. It is a high level qualifier command which may appear anywhere in the job. Each occurrence of !ASSIGN must start on its own input line. The syntax is !ASSIGN *name string* or !ASSIGN *name* !< *string* !> and the defined *string* is substituted into the job where $*name* appears. *string* is the rest of the line and may include blanks. If !< !> encloses *string*, *string* may extend over several lines, which are concatenated. For example !ASSIGN TVS xfa1(Treat) ... ... $TVS.geno ... is interpreted as ... xfa1(Treat).geno ... **Restrictions:** • a maximum of 50 assign strings may be defined. • the combined length of all strings is 5000 characters. • *name* may have up to 8 characters but should not begin with a number (see command line arguments). • dollar substitution occurs before most other high level actions. ASSIGN strings and commandline arguments may substitute into a !CYCLE line. • I, J, K and L are reserved as names referring to items in the !CYCLE list and should therefore not be used as names of an ASSIGN string.
!CYCLE [!SAMEDATA] *list*	There is now an extra qualifier with !CYCLE, !SAMEDATA. Putting !SAMEDATA on the (leading) !CYCLE line makes ASReml read the data (and .grr) file in the first CYCLE and hold it in memory for use in subsequent cycles. This is advantageous when the data/.grr file is large and there are many CYCLEs to execute where the model changes, but the data/.grr file doesn't.

1.2 Improvements to analysis

High level qualifiers

qualifier	action

!FOR *forlist* !DO *command*	The !FOR ... !DO ... command is intended to simplify coding when a series of similar lines are required in the command file which differ in a single argument. The list of arguments is placed after !FOR and the command is written after !DO with $S indicating where the argument is to be inserted. *list* may be an assign string since they are processed before the !FOR statement is expanded. Furthermore, if *list* is entirely integer numbers, *i*:*j* notation can be used.
	For example
	`!ASSIGN Markern 35 75 125`
	`!ASSIGN Markers M35 M75 M125`
	`!FOR $Markern !DO !MBF mbf(Geno,1) markers.csv !key 1 !RFIELD$S !RENAME M$S`
	`... ... !r $Markers`
	is expanded to
	`!MBF mbf(Geno,1) markers.csv !key 1 !RFIELD 35 !RENAME M35`
	`!MBF mbf(Geno,1) markers.csv !key 1 !RFIELD 75 !RENAME M75`
	`!MBF mbf(Geno,1) markers.csv !key 1 !RFIELD 125 !RENAME M125`
	`... ... !r M35 M75 M125`
	
	The aim here is to generate the 3 !MBF statements required to extract markers 35, 75 and 125 from the marker file *markers.csv*. The names of model terms must begin with a letter, hence the marker names are the letter M followed by the position number. Alternatively !RFIELD*lettersinteger* is interpreted as !RFIELD *integer* so the !FOR statement can be written even more concisely as
	`!FOR $Markers !DO !MBF mbf(Geno,1) markers.csv !key 1 !RFIELD$S !RENAME M$S`
	without the need to assign Markern. Now, to add another marker to the model, one can just add the marker integer to the ASSIGN statement.
	
	Restriction: *list* and *command* are both limited to 200 characters.

!IF *string1 == string2 text*	One form of the IF statement is
	`!IF string1 == string2 !ASSIGN M1 brt DamAge` which makes the !ASSIGN statement active if string1 is the same as string2. Note that there need to be spaces before and after == to avoid confusion with the strings. This has been used when performing a large number of bivariate analyses with trait specific fixed effects being fitted. So
	`...`
	`!IF $1 == wwt !ASSIGN M1 brt DamAge`
	`!IF $1 == ywt !ASSIGN M1 brt`
	`!IF $1 == fwt !ASSIGN M1 DamAge`
	`!IF $2 == wwt !ASSIGN M2 brt DamAge`
	`!IF $2 == ywt !ASSIGN M2 brt`
	`!IF $2 == fwt !ASSIGN M2 DamAge`
	
	`...`
	`$1 $2 ~ Trait at(Trait,1).($M1) at(Trait,2).($M2)`

1.2.6 Factor effects with large Random Regression models

One use of the GRM matrix is to allow more computationally efficient fitting of random regression models associating $\boldsymbol{u}$, a vector of f factor effects with $\boldsymbol{v}$ a vector of m regression effects through the model $\boldsymbol{u} = \boldsymbol{M}\boldsymbol{v}$ where the matrix $\boldsymbol{M}$ contains m regressor variables for each of the f levels of the factor. Direct fitting of the regression effects is facilitated by using the my basis function (`mbf` function) associating the regressor variables to the levels of the factor, essentially fitting $\boldsymbol{Z}\boldsymbol{M}\boldsymbol{v}$ where $\boldsymbol{Z}$ is the design matrix linking observations to the levels of the factor. But if m is much bigger than f, it is more computational efficient to fit an equivalent model $\boldsymbol{Z}\boldsymbol{u}$ with a variance structure for $\boldsymbol{u}$ based on $\boldsymbol{M}\boldsymbol{M}'$. ASReml can read the matrix $\boldsymbol{M}$ associated with a factor and group of regressor variables from a `.grr` file, construct a GRM matrix ($\boldsymbol{G} = \boldsymbol{M}\boldsymbol{M}'/s$), fit the equivalent model and report both factor and regressor predictions. One common case of this model is when $\boldsymbol{u}$ represents genotype effects, the regressors represent SNP marker counts (typically 0/1/2) and $\boldsymbol{v}$ are marker effects.

The `.grr` file is specified after any pedigree file and before the data file (with any other GRM files). There may only be one `.grr` file. It is assumed to contain a row for each level of the factor, each row containing m regressor values. Optionally the factor level name associated with the i-th row can be included before the relevant regressor values. Also a heading row might include a name for each field/regressor variable. Superfluous fields before the factor or regressor fields can be skipped and superfluous rows before the regressor information can be skipped.

The syntax for specifying and reading the `.grr` file is

M`.grr [!CSKIP` c_1`]` ***Factor*** `[`f`] [!NOID] [!CSKIP` c_2`]` ***Regressors*** `[`m`] [!NONAMES] [!SKIP` s`]` where

M`.grr` is the name of the file to be read, `!CSKIP` c_1 indicates c_1 fields are to be skipped before the factor identifiers are read,

Factor is the name of the variable in the data that is associated with the regressors,

f sets the maximum number of levels (default 1000) of *Factor* with regressor data; ASReml will count the actual number,

`!NOID` indicates that the factor identifiers are not present in the `.grr` file,

`!CSKIP` c_2 indicates c_2 fields are to be skipped before the regressor variables are read,

Regressors is the name for the set of regressor variables,

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

`!SKIP` s specifies how many lines are to be skipped before reading the regressor data,

`!NONAMES` indicates there is no line containing the individual names of the regressor variables; otherwise names are taken from the first (non-skipped) line in the file.

1.2 Improvements to analysis

If the factor identifiers are not present (`!NOID`), ASReml assumes that the order of the factor classes in the data file matches the order in the `.grr` file. If the factor identifiers are present, ASReml uses the identifiers obtained from the `.grr` file to define the order of the factor classes when the data is read; any extra identifiers in the data not in the `.grr` file are appended at the end of the factor level name list. If `!NOID` is set, identifiers in the `.grr` file are not needed and if present should be skipped using `!CSKIP`.

Values are typically `TAB`, `COMMA` or `SPACE` separated but may be packed (no separator) when all values are integers 0/1/2. Missing values in the regression variables may be represented by `*`, `NA`. Invalid data is also treated as missing. Missing values are replaced by the mean of the respective regressor. Alternative missing data methods that involve imputation from neighbouring markers have not been implemented.

Some general qualifiers are:
`!SAVEGIV` instructs ASReml to write the $\boldsymbol{G}$ matrix in `.dgiv` format,
`!PSD` s declares that the derived variance matrix may have up to s singularities,
`!PEV` requests calculation of Prediction Error Variance of marker effects which are reported in the `.mef` file. Calculation of Prediction error variances is computationally very expensive,
`!CENTRE` $[c]$ requests ASReml to centre the regressors at c if c is specified else at the individual regressor means; otherwise the $\boldsymbol{G}$ matrix is formed from uncentered regressors. Note that centring introduces a singularity in the G matrix and `!PSV` s will need to be set.

Other qualifiers relate specifically to whether the regressors are markers. Markers are typically coded 0/1/2 being counts of the minor allele. However, if they are imputed, they will take real values between 0 and 2. Since marker files may be huge,
`!SMODE` b sets the storage mode for the regressor data, indicating whether it is marker data: $b = 2$ sets 2bit storage for strictly 0/1/2 marker data, $b = 8$ (the default) sets 8bit storage useful for marker data with imputed values having 2 digits after the decimal, $b = 16$ sets 16bit storage useful for marker data with imputation with more than 2 digits and $b = 32$ sets 32bit real storage and should be used for non-marker data,
`!RANGE` l h indicates the marker scores range $l : h$ and are to be transformed to have a range 0:2,
`!GSCALE` s, controls the scaling of the GRM matrix. If unspecified $s = \Sigma 2p(1-p)$ is used for marker data, $s = 1$ for non marker data (`!SMODE 32`). Scaling is often used with centred marker data to scale the $\boldsymbol{MM}'$ matrix so that it is a genomic relationship matrix.

Example

```
 !WORK 1
Nassau Clone Data
 Nfam 71 !A
 Nfemale 26 !A
 Nmale 37 !A
 Clone !A 860
 rep 8
```

```
 iblk 80
 culture !A
 DBH6

snpData.grr Clone Marker

nassau.csv !MAXIT 30 !SKIP 1 !DFF -1
DBH6 ~ mu culture/rep !r grm1v(Clon) 0.27 Clone 0.15 rep.iblk 0.31
```

where `snpData.grr` is first used to declare Clone identifiers (taken from the first field) in the correct order, and then contains the marker scores; it looks like

```
Genotype,0-10024-01-114,0-10037-01-257,0-10040-02-394,...
140099,2,2,1,2,2,2,2,2,2,1,2,1,2,1,1,2,1,2,2,2,2,2,1,2...
141099,2,2,0,0,2,2,1,2,2,1,2,1,2,2,0,2,2,2,2,1,2,2,1,1...
...
547853,2,2,1,2,2,2,1,2,2,0,2,1,2,2,2,2,2,2,2,1,2,...
547966,2,2,1,1,1,2,0,2,2,1,2,2,2,2,2,2,2,2,2,1,2,...
548082,2,2,1,2,2,2,1,2,1,2,2,1,2,2,1,2,2,2,2,1,2,...
```

The primary output follows:

```
Nfam 71 !A
 Nfemale 26 !A
 Nmale 37 !A
 Clone  !A 860
 MatOrder 914 !A
 rep 8 !A
 iblk 80 !A
 prop 1 !A
 culture 2 !A
 treat 2 !A
 measure 1 !A
 CWAC6 !M-9
 Parsing: snpData.grr Clone
 Class names for factor "Clone" are initialized from the .grr file.
 GRR Header line begins: Genotype,0-10024-01-114,0-10037-01-257,0
        4854 Marker labels found
Marker labels 0-10024-01-114 ... UMN-CL98Contig1-
 Notice: The header line indicates there are 4854 regressors in the file.
 Notice: SNP data line begins: 140099,2,2,1,2,2,2,2,2,2,1,2,1,2,1,1,
 Notice: Markers coded -9 treated as missing.
 Marker data [0/1/2] for 923 genotypes and 4854 markers read from snpData.grr
      160414 missing Regressor values (  3.6%) replaced by column average!
        Regressor values ranged 0.00 to 2.00
        Regressor Means ranged  1.00 to 2.00
          Sigma2p(1-p) is   1057.12558
 GIV1  snpData.grr      923       9     -946.27
 QUALIFIERS: !MAXIT 30 !SKIP 1 !DFF -1
 QUALIFIER: !DOPART    2 is active
```

1.2 Improvements to analysis

```
Reading nassau_cut_v3.csv  FREE FORMAT skipping     1 lines

Univariate analysis of HT6
Summary of 6399 records retained of 6795 read

 Model term          Size #miss #zero    MinNon0    Mean       MaxNon0  StndDevn
  1 Nfam               71     0     0       1    36.3379         71
  2 Nfemale            26     0     0       1    12.8823         26
  3 Nmale              37     0     0       1    15.2285         37
 Warning: More levels found in Clone  than specified
  4 Clone             926     0     0       1   464.6765        926
 Warning: Fewer levels found in MatOrder  than specified
  5 MatOrder          914     0     0       1   432.5760        860
  6 rep                 8     0     0       1     4.4837          8
  7 iblk               80     0     0       1    40.1164         80
  8 tree                      0     0 1.0000      7.473       14.00       4.018
  9 row                       0     0 1.0000      28.52       56.00       16.09
 10 col                       0     0 1.0000      10.50       20.00       5.760
 Warning: Fewer levels found in prop  than specified
 11 prop                2     0     0       1     1.0000          1
 12 culture             2     0     0       1     1.4945          2
 13 treat               2     0     0       1     1.4945          2
 Warning: Fewer levels found in measure  than specified
 14 measure             2     0     0       1     1.0000          1
 15 SURV                     0     6 1.0000     0.9991      1.0000      0.3061E-01
 16 DBH6                     4     0 0.3000E-01  11.29       18.80       2.400
 17 HT6            Variate   0     0  76.20      838.6       1286.       163.6
 18 HT8                     83     0  91.44      1148.       1576.       170.6
 19 CWAC6                 3167     0  97.54      301.3       542.5       52.26
 20 mu                       1
 21 culture.rep             16 12 culture    :   2   6 rep            :    8
Warning: GRM matrix is too SMALL
 22 grm1(Clone)       923
 23 rep.iblk                640  6 rep       :   8   7 iblk           :   80
Forming     2508 equations:  19 dense.
Initial updates will be shrunk by factor    0.316
Notice: LogL values are reported relative to a base of -30000.000
Notice:     11 singularities detected in design matrix.
  1 LogL=-2845.97     S2=  8956.5       6390 df
  2 LogL=-2799.30     S2=  8568.1       6390 df
  3 LogL=-2759.03     S2=  8131.3       6390 df
  4 LogL=-2741.99     S2=  7766.2       6390 df
  5 LogL=-2741.40     S2=  7702.9       6390 df
  6 LogL=-2741.40     S2=  7700.1       6390 df

         - - - Results from analysis of HT6 - - -
Akaike Information Criterion    65490.79 (assuming 4 parameters).
Bayesian Information Criterion  65517.84

Model_Term                           Gamma         Sigma   Sigma/SE   % C
rep.iblk               IDV_V  640  0.307856       2370.52      13.00   0 P
grm1(Clone)            GRM_V  923  0.275656       2122.58       5.82   0 P
Clone                  IDV_V  926  0.152554       1174.68       6.08   0 P
Residual               SCA_V 6399  1.000000       7700.10      49.64   0 P
```

```
                                Wald F statistics
    Source of Variation         NumDF              F-inc
 20 mu                              1           0.11E+06
 12 culture                         1            2615.96
 21 culture.rep                     6              30.44
 23 rep.iblk                      640 effects fitted
 22 grm1(Clone)                   923 effects fitted
  4 Clone                         926 effects fitted (      66 are zero)
        78  possible outliers: see .res file
```

Notes:

- of 926 clones identified, 860 have data and 923 have genomic data.
- The `.res` file contains additional details about the analysis including a listing of the larger marker effects. All marker effects are reported in the `.mef` file.
- Particular columns of the `.grr` data can be included in the model using the `grr(Factor,i)` model term where and i specifies which (number) regressor variable to include.

```
Listing of the larger marker effects
      368  0-12761-01-121     1.40736       0.00000
      617  0-14383-01-111     1.26081       0.00000
      777  0-15417-01-138    -1.25597       0.00000
     1246  0-18644-02-210     1.22522       0.00000
     1903  0-6963-01-202     -1.24800       0.00000
     2102  0-8683-02-432      1.15496       0.00000
     2445  2-1563-02-244     -1.35181       0.00000
     2497  2-2167-01-413     -1.21339       0.00000
     3180  2-8668-03-42      -1.21629       0.00000
     3521  CL1577Contig1-03  -1.15833       0.00000
     3802  CL2573Contig1-03   1.17005       0.00000
     4195  CL595Contig1-01-  -1.19330       0.00000
     4351  UMN-1397-01-416   -1.34916       0.00000
```

1.2.7 Functions of variance components

Two new functions have been introduced to allow multiplication of parameters and taking square roots of parameters and calculating standard errors of these functions.

`S` *label i:j* when *i:j* are assumed positive variance parameters, inserts components which are the SQRT of components *i:j*,

`X` *label i*k* inserts a component being the product of components i and k.

`X` *label i:j*k* inserts $j - i + 1$ components being the products of components $i : j$ and k.

`X` *label i:j*k:l* inserts a set of $j-i+1$ components being the pairwise products of components $i:j$ and $k:l$.

The multiply option (`X`) allows a correlation in a `CORUV` structure to be converted to a covariance. The `SQRT` option allows conversion of `CORGH` to `US`, provided the dimension is moderate (say < 10).

One can also convert CORUH and XFA to US using

`V` *label i:j* where $i:j$ spans an XFA variance structure, inserts the US matrix based on the XFA parameters.

`V` *label i:j* where $i:j$ spans a CORUH variance structure, inserts the US matrix based on the CORUH parameters.

When ASReml reads back the variance parameters from the `.asr` file, the parameters are given a name based on the random linear model term. The parameters in the R structures are effectively given a name `Residual`. The individual variance parameters associated with the linear model term can be specified by number, or sequence of numbers (`n:m`) by appending these in square braces after the linear model term, for example `C.Trait[3]` or `Residual[4:6]`. Users may contract names if they do not cause ambiguity, for example `Sire.Trait` might be contracted to `Sire` if there are no other random terms including `Sire`. If the user is in doubt of the name or number of a parameter then running the program with `VPREDICT !DEFINE` and a blank line will construct a `.pvc` file with the names and numbers of parameters identified.

Critical change For generalised linear models in ASReml 4 the `.pvc` file reports and numbers, for completeness, a residual or dispersion parameter both when the parameter is estimated or fixed. By contrast `ASReml 3` does not report or number if the parameter is fixed, by default, at 1. Hence the parameters might be numbered differently in ASReml 4 and ASReml 3.

1.2.8 A detailed example

The following example for a **bivariate sire model** is a little more complicated. The job file `bsiremod.as` contains

```
...
coop.fmt

ywt fat ~ Trait Trait.(age c(brr) sex sex.age) !r Trait.sire  !f Tr.grp
1 2 1
0 0 ID
Trait 0 US * !GP
Trait.sire 2
Trait 0 US * !GP
sire 0 ID
```

1.2 Improvements to analysis

```
VPREDICT !DEFINE
F phenvar Residual + Sire;Trait                              # 1:3 + 4:6
F addvar sire * 4                                            # 4:6 * 4
H heritA addvar[1] phenvar[1]                                # 10 7
H heritB addvar[3] phenvar[3]                                # 12 9
R phencorr phenvar                                           # 7 8 9
R gencorr addvar                                             # 4:6
```

The relevant lines of the `.asr` file are

```
 Model_Term                          Sigma         Sigma   Sigma/SE   % C
 Residual                  8140 effects
 Residual              US_V  1  1   23.2055       23.2055      44.44   0 P
 Residual              US_C  2  1   2.50402       2.50402      18.56   0 P
 Residual              US_V  2  2   1.66292       1.66292      32.82   0 P
 Trait.sire            US_V  1  1   1.45821       1.45821       3.66   0 P
 Trait.sire            US_C  2  1  0.130280      0.130280       1.92   0 P
 Trait.sire            US_V  2  2  0.344376E-01  0.344376E-01   2.03   0 P
```

Numbering the parameters reported in `bsiremod.asr` (and `bsiremod.vvp`)

1 error variance for `ywt`
2 error covariance for `ywt` and `fat`
3 error variance for `fat`
4 sire variance component for `ywt`
5 sire covariance for `ywt` and `fat`
6 sire variance for `fat`

then

`F phenvar Residual + Trait.sire` or
`F phenvar Residual + sire;Trait` or `F phenvar 1:3 + 4:6`

creates new components **7** = **1**+**4**, **8** = **2**+**5** and **9** = **3**+**6**,

`F addvar sire;Trait * 4` or `F addvar 4:6 * 4`

creates new components **10** = **4** $\times$ 4, **11** = **5** $\times$ 4 and **12** = **6** $\times$ 4,

`H heritA addvar[1] phenvar[1]` or `H heritA 10 7`

forms **10** / **7** to give the heritability for `ywt`,

`H heritB addvar[3] phenvar[3]` or `H heritB 12 9`

forms **12** / **9** to give the heritability for `fat`,

`R phencorr phenvar` or `R phencorr 7 8 9`

forms **8** $/\sqrt{\mathbf{7} \times \mathbf{9}}$, that is, the phenotypic correlation between `ywt` and `fat`,

`R gencorr addvar` or `R gencorr 4:6`

forms **5** $/\sqrt{\mathbf{4}\times\mathbf{6}}$, that is, the genetic correlation between `ywt` and `fat`.

The resulting `.pvc` file contains:

```
Residual                      8140 effects
  1 Residual;Residual                        V  1  1      23.2055        0.522176
  2 Residual;Residual                        C  2  1      2.50402        0.134915
  3 Residual;Residual                        V  2  2      1.66292        0.506679E-01
  4 Trait.sire                               V  1  1      1.45821        0.398418
  5 Trait.sire                               C  2  1     0.130280        0.678542E-01
  6 Trait.sire                               V  2  2    0.344376E-01     0.169643E-01
  7 phenvar  1                 24.664       0.64250
  8 phenvar  2                 2.6343       0.14763
  9 phenvar  3                 1.6974       0.52366E-01
 10 addvar  4                  5.8328        1.5926
 11 addvar  5                 0.52112       0.27170
 12 addvar  6                 0.13775       0.67799E-01
 13 heritA       = addvar    10/phenvar    7=          0.2365    0.0612
 14 heritB       = addvar    12/phenvar    9=          0.0812    0.0394
 15 phenco  2  1 = phenv  8/SQR[phenv  7*phenv  9]=    0.4071    0.0183
 16 gencor  2  1 = addva 11/SQR[addva 10*addva 12]=    0.5814    0.2039
Notice: The parameter estimates are followed by
         their approximate standard errors.
```

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

1.3 Modifications and improvements to output

1.3.1 Output formats in the .asr file

The general aim has been to make as few changes as possible to output formats. However, there are some minor changes with this release. These are likely to effect users who have developed in-house routines to parse ASReml output files. Some field widths have been changed to reduce the frequency of fields running together. In particular, the header line for the table of variance components has been changed to

```
Model_Term                                Gamma          Sigma    Sigma/SE    % C
```

or

```
Model_Term                                Sigma          Sigma    Sigma/SE    % C
```

and the contents of the leading fields modified to better identify the particular components. These now show the model term, the component (of an interaction) and the variance structure, the parameter type and the indexing or size information in a more consistent manner than before.

1.3.2 Writing out a design matrix

The new qualifier `!DESIGN` on the datafile line causes ASReml to write the design matrix, not including the response variable, to a `.des` file. It allows ASReml to create the design matrix required by the `VCM` process, see Section 1.2.1. For example, using a control file `vcmdes.as` containing

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

and a data file `vcmdes.asd` containing

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

then the file `vcmdes.des` will be generated which contains the values used in fitting the variance model for the HuynhFeldt model given in Section 1.2.1.

1.3.3 XML output

The primary tables reported in the `.asr` file and key output from `.pvs` and `.sln` files are written to `.xml` file in xml format. Output is presented in the order of computation. The first block written is a `.asr` block includes start and finish times, the data summary, the iteration sequence summary and information criteria, then from the `.pvs` file the tables and associated information, then the summary of estimated variance structure parameters from the `.asr` file, then information from the `.sln` file then finally the Wald F statistics and completion information from the `.asr` file. The process is repeated for each cycle of analysis. The intended use of this file is by programs written to parse ASReml output. For further details, including the status of intended future developments, please contact `support@vsni.co.uk`.

1.3.4 Writing output to a separate folder

`!OUTFOLDER` [*path*] allows most of the output files to be written to a folder other than the working folder. This qualifier must be placed on the top command line as it needs to be processed before any output files are opened. Most files produced by ASReml have a filename structure

<*basename*><*subname*>.<*extension*>

where <*subname*> is a command line argument value. If `!OUTFOLDER` is specified without *path*, the output filename pattern becomes

<*basename*><*subname*>/<*basename*>.<*extension*>

If *path* is specified, the output filename pattern becomes

<*path*>/<*basename*><*subname*>.<*extension*>

There are a few files written by ASReml that do not follow this naming pattern, for example, `ainverse.bin` and `asrdata.bin`. These remain unchanged, that is, they are not written to the output folder.

1.3.5 Prediction using two-way interaction effects

In some cases we wish to calculate from two way interaction effects, bc_{ij} say, effects for one of the factors, B say, that are a weighted sum averaged over the c levels of C, ie. $b_i = \sum_{j=1}^{c} bc_{ij} w_j$.

```
TPREDICT C !AVE B weights !ONLYUSE B.C
```

allows this to be produced more computationally efficiently than it would be using `PREDICT`. For example,

```
TPREDICT Animal !AVE Trait 2.1 1.2 -7.4 !ONLYUSE Trait.Animal
```

Part of the motivation for this is the calculation of selection indices. The index coefficients are typically derived as $w = a' G_{om} G_{mm}^{-1}$ where G_{mm} is the variance matrix for the measured traits (corresponding to C in the example), G_{om} is the genetic covariance matrix between the objective traits and the measured traits, and a is the vector of economic values for the objective traits. The results are given in a `.sli` (selection index) file. This directive should be placed after the model specification.

1.3.6 Solution (.sln) file

A header

```
Model_Term                Level                Effect    seEffect
```

has been added. Note that extra signifcant digits are reported when `!SLNFORM` is set, and expanded labelling of the levels in interactions is used because field width is no longer restricted.

1.4 Functional specification of the mixed model

The main development in Release 4 is the implementation of a functional model specification. This is described in detail in Chapter 2. The distinguishing feature of this specification is that the variance models for individual terms are specified by applying variance model functions to the terms in the model definition lines. This is in contrast to the structural specification of versions prior to Release 4 in which the variance models were only imposed after the mixed model definition line(s).

The functional specification is described through a series of examples of increasing complexity, using the Nebraska Intrastate Nursery (NIN) data (Chapter 3 of the User Guide) for demonstration. A more comprehensive set of examples is presented in Chapter 3.

2 Functional specification of the mixed model

2.1 Introduction

Prior to Release 4, variance structures were applied to random model terms and the residual error term by specifying these structures after the mixed model had been specified. In Release 4, a new functional specification of the variance model has been implemented in which variance structures can be specified by applying variance model functions directly to terms in the model definition line(s). This offers a more concise and less error-prone alternative to the structural specification and mirrors the implementation in ASReml-R. An understanding of the new specification should facilitate free movement between the standalone and R platforms.

The functional specification is introduced for the random model terms in Section 2.4 and the residual error term in Section 2.5. In Section 2.7 the NIN data is used for demonstration, with coding included for both specifications to allow direct comparison between the existing and new code. More advanced aspects of the functional specification are presented after the example.

2.2 Typographic convention

A box system is used to assist existing users in migrating from the structural specification to the functional specification that is new with Release 4:

Boxes like this are used throughout this chapter to present the structural specification corresponding to the particular aspect of the functional specification under discussion.

2.3 The theory

From Chapter 2 of the ASReml User Guide, the linear mixed model can be written as

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

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

2.3.1 Sigma parameterization of the linear mixed model

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

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

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

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

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

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

Example 2.1 A simple example

Consider a one-way classification comprising a single random effect $\boldsymbol{u}$, and a residual error term $\boldsymbol{e}$. The two random components of this model, namely $\boldsymbol{u}$ and $\boldsymbol{e}$, are 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

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

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

□

2.3.2 Partitioning the fixed and random model terms

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

2.3.3 G structure for the random model terms

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

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

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

The default assumption is that each random model term generates one component of this direct sum (then $b' = b$ and $\text{var}(\boldsymbol{u}_i) = \boldsymbol{G}_i$ for $i = 1 \dots b$). This means that the random effects from any two distinct model terms are uncorrelated. However, in some models, one component of $\boldsymbol{G}$ may apply across several model terms, for example, in random coefficient regression where the random intercepts and slopes for subjects are correlated, see 2.4.2. To accommodate these cases, one component of $\boldsymbol{G}$ may apply across several model terms (then $b' < b$). In some other (less likely but possible) cases, we may wish to separate one model term over several independent parts (then $b' > b$), see Section 2.4.2.

Example 2.2 Variance components mixed models

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

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

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

$$\text{var}(\boldsymbol{y}) = \sum_{i=1}^{b} \sigma^2_{u_i} \boldsymbol{Z}_i \boldsymbol{Z}_i^\intercal + \sigma^2_e \boldsymbol{I}_n. \tag{2.5}$$

□

2.3.4 Partitioning the residual error term

As for the fixed and random model terms, it is often useful or appropriate to consider a partitioning of the vector of residual errors $\boldsymbol{e}$ according to some conditioning factor. We

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

2.3.5 R structure for the residual error term

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

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

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

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

Typically a variance structure is specified for each random model term and often more complex models than the simple IID model are specified. ASReml offers a wide range of variance models to choose from. They are listed in Table 2.5 and are described in more detail in the following sections.

2.3.6 Gamma parameterization for the linear mixed model

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

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

for γ_{g_i} being the ratio of the variance component for the random term $\boldsymbol{u}_i$ relative to error variance, that is, $\gamma_{g_i} = \sigma_{u_i}^2 / \sigma_e^2$. In this case ASReml calculated a simple estimate of σ_e^2 and initial values for the iterative process were specified in terms of the ratios γ_{g_i} rather than in

terms of the variance components $\sigma^2_{u_i}$. It was often easier to specify initial values in terms of these ratios rather than the variance components which is why this approach was adopted. Where $\boldsymbol{R}_v(\boldsymbol{\sigma}_r)$ can be written as a scaled correlation matrix, that is, $\boldsymbol{R}_v(\boldsymbol{\sigma}_r) = \sigma^2_e \boldsymbol{R}_c(\boldsymbol{\gamma}_r)$, this suggests the alternative specification of (2.2)

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

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

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

which we will refer to as the *gamma parameterization*, and the individual variance structure parameters in $\boldsymbol{\gamma}_g$ and $\boldsymbol{\gamma}_r$ will be referred to as *gammas*. ASReml switches between the sigma and gamma parameterizations for estimation. This is discussed further in Section 2.8.

2.3.7 Parameter types

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

Parameterization prior to Release 4 When ASReml was developed, (2.2) and (2.8) were combined into

$$\begin{bmatrix} \boldsymbol{u} \\ \boldsymbol{e} \end{bmatrix} \sim N\left(\begin{bmatrix} \boldsymbol{0} \\ \boldsymbol{0} \end{bmatrix}, \theta \begin{bmatrix} \boldsymbol{G}(\boldsymbol{\gamma}) & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{R}(\boldsymbol{\phi}) \end{bmatrix} \right) \tag{2.9}$$

where θ was referred to as a scaling parameter and was used in specifying the variance model. The way θ was treated depended on the way $\boldsymbol{R}(\boldsymbol{\phi})$ was defined. If $\boldsymbol{R}(\boldsymbol{\phi})$ included variance parameters then, to avoid problems in estimability and for convenience, we set $\theta = 1$. In this case, ASReml used the sigma parameterization and estimated sigmas were reported. Alternatively, if $\boldsymbol{R}(\boldsymbol{\phi})$ was defined to be a correlation matrix (diagonal elements equal to 1) then the gamma parameterization was used and the estimated gammas were reported. In this case the estimated sigmas were also reported.

2.4 Applying variance structures to random model terms

The random model terms $\boldsymbol{u}_i$ in $\boldsymbol{u}$ define the random effects and associated design matrices, $\boldsymbol{Z}_i \in \boldsymbol{Z}$, but additional information is required before the model can be fitted. This extra step involves defining the G structure for each term. In general, there are two different approaches to this problem which we refer to as the *structural approach* and the *functional approach*. The structural approach applies variance models to individual terms *after* the

random model has been defined. This has been the approach of ASReml until Release 4 and gives a step-by-step approach to building up the full model that is straightforward but not concise. In contrast, the functional approach that is new to Release 4, uses functions to directly apply variance models to the components of the random model term to produce a *consolidated model term* that simultaneously defines both the design matrix ($\boldsymbol{Z}_i$) and variance model ($\boldsymbol{G}_i$).

2.4.1 Process to define a consolidated model term

Consider a linear model term `column.row` comprising the interaction between the single model terms column and row. We refer to `column.row` as a *compound model term*. If the variance structure for `column.row` is the direct product of two matrices, the first of which is an IID variance structure that is a scaled identity matrix with dimension equal to the number of levels of the factor `column` and the second of which is a matrix with dimension equal to the number of levels of the factor `row` and with elements representing a first order autoregressive correlation structure `AR1`, then we represent this by the consolidated model term `idv(column).ar1(row)`. This specifies a two-dimensional separable spatial variance structure for `column.row` but with spatial correlation in the `row` direction only. A consolidated model term is therefore comprised of component terms, each with a variance model function applied to give the required direct product form of the variance structure. Table 2.1 demonstrates how to build consolidated terms in ASReml for a small selection of examples. The linear model term (single or compound) is first identified (*column 2*) and the individual components that identify the dimension of the individual matrices used in forming the direct product variance structure are then written down (*column 3*). Note that in the simplest cases there is only one component. The variance structure associated with each component has a structure name (*column 4*) and a corresponding variance model function name (*column 5*) giving the associated component variance structures (*column 6*). The consolidated model term is the term presented in the final column of the table. In contrast, in ASReml 3 the linear model terms are defined on the model line and subsequently a G structure line is given for each linear model term which specifies the component terms and their associated structures. The simplest form of a consolidated model term is a single model term with a variance model function applied, eg. `idv(repl)` in Table 2.1, and the next simplest is a compound model term with a variance model function applied, eg. `idv(A.B)` in Table 2.1.

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

- variance model functions can be applied to single model terms (see example **1** in Table 2.1), the components in compound model terms (examples **4** to **6**) and constructed linear model functions of variables (example **2**),
- variance model functions can also be applied to compound model terms (example **3**)
- variance model functions *cannot* be applied to expandable model terms, for example, to
  - `A*B` which expands to `A B A.B`

- `A/B` which expands to `A A.B`
- `at(A,i,j).B` which expands to `at(A,i).B at(A,j).B`

- only one component of a variance structure for a compound model term may include a variance parameter, the other components must be correlation structures (no associated variance parameter). This is due to the identifiability issues that occur when multiple variance structures are specified. This is explained in NIN example **3a**, see Section 2.7. The defined variance function may be homogeneous (name ending in `v`) or heterogeneous variance (name ending in `h`). This is discussed in detail in Section 2.4 of the User Guide.

Some common variance functions are defined in Table 2.2; the full range of variance model functions and their detailed definition is presented in Table 2.5.

Table 2.1: Building consolidated model terms in ASReml

	linear model term (*type of term*)	component(s)	variance structure name	variance model function name	covariance component	consolidated model term
1	`repl` *single*	`repl`	IDV	`idv()`	`idv(repl)`	`idv(repl)`
2	`fac(x)` *single*	`fac(x)`	EXPV	`expv()`	`expv(fac(x))`	`expv(fac(x))`
3	`A.B` *compound*	`A.B`	IDV	`idv()`	`idv(A.B)`	`idv(A.B)`
4	`column.row` *compound*	`column` `row`	IDV AR1	`idv()` `ar1()`	`idv(column)` `ar1(row)`	`idv(column).ar1(row)`
5	`site.variety` *compound*	`site` `variety`	DIAG ID	`diag()` `id()`	`diag(site)` `id(variety)`	`diag(site).id(variety)`
6	`Trait.animal` *compound*	`Trait` `animal`	US NRM	`us()` `nrm()`	`us(Trait)` `nrm(animal)`	`us(Trait).nrm(animal)`

2.4.2 Modelling a single variance structure over several model terms

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

Table 2.2: List of common variance model functions, their type (correlation or variance), the form of the variance matrix generated ($\boldsymbol{C}$ for correlation, $\boldsymbol{V}$ for variance matrix), and a brief description. Parameters $\sigma_i^2 > 0$ are variances, $-1 < \rho_i < 1$ are correlations. Subscipt c denotes parameter held in common across all rows/columns.

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

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

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

Example 2.3 Random coefficient regression

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

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

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

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

Using the structural specification, the model term is written as
```
!{ Animal age.Animal !} ...
```
and the G structure lines as
```
Animal 2
2 0 US
0.3 0.01 0.2
Animal 0 ID
```
The idea was to define a variance structure that is associated with effects in the linear model starting at the first term specified and exactly spanning the whole set of terms given inside the `!{ !}` brackets.

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

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

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

```
str(Mpar and(Fpar) id(1).idv(Mpar))
```

overlays the `Mpar` and `Fpar` design matrices and allows the resulting effects to be modelled with a single variance parameter.

□

Example 2.4 Fitting a genetic covariance between direct and maternal effects

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

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

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

Using the structural specification, the model term is written as

```
... !{ Trait.animal at(Trait,1).dam !} ...
```

and the G structure lines as

```
Trait.animal 2
3 0 US
0.3
0.01 0.2
0.01 0.01 0.2
animal 0 NRM #was AINV in ASReml 3
```

□

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

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

will do this.

2.5 Applying variance structures to the residual error term

In Release 4 the residual error term is also defined using a consolidated model term, and it now appears after a `residual` statement that has been introduced to specify the associated variance structure. We give five examples. Firstly, for the default situation of IID residual errors (*1. in box on page 34*), the error model definition line would be

```
residual idv(units)
```

This second example would specify a separable autoregressive spatial model of order 1 (*AR1×AR1)* for the observations from a trial arranged in a rectangular array indexed by the data variables `column` and `row` (*2. in box on page 34*). To apply this variance structure the observations would need to cover the whole grid, but it would not be necessary to pre-order the data file as rows within columns as ASReml uses the information in `column` and `row` to

put the observations into the appropriate row within column order:

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

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

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

would be an equivalent coding for the $AR1 \times AR1$ model using the dimensions of the factors rather than the factor names (*3. in box on page 34*). In this case the data records *would need to be* sorted in the order rows within columns because ASReml does not have the information needed to reorder the data internally.

The fourth example assumes variance heterogeneity among the data observations, that is, that the three groups comprising observations 1...23, 24...50, 51...70 have unequal variances (*4. in box on page 34*):

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

The fifth and final example is the default residual variance in a multivariate analysis (*5. in box on page 34*). Specifying `units` as the first component is crucial as ASReml extracts the trait values by trait within unit:

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

In the structural specification the R structure is specified by a *Variance header line* consisting of three numbers s c g, followed by s sets of c variance structure lines. s is the number of sections (see below), c is the number of components in the R structure for each section, and g is the number of model terms with G structure definitions (described in the preceding section).

The following examples match the examples above as per the common italicized model descriptions. This is the way the model definition has been applied in versions of ASReml prior to Release 4

1. IID residual errors
no residual error term specified

2. separable autoregressive spatial model of order 1 (AR1×AR1)

```
1 2 0 #1 section, 2 variance terms, 0 G structures
rep rep AR1 .1 #the second 'rep' names the data column for sorting
plot plot AR1 0.1
```

3. an equivalent coding for the AR1×AR1 model using the dimensions of the factors rather than the factor names

```
1 2 0
3 0 AR1 .1
23 0 AR1 0.1
```

4. variance heterogeneity among the data observations

```
3 1 0
23 0 ID #default estimate residual variance for all 3 sections
27 0 ID
20 0 ID
```

5. default residual variance in a multivariate analysis

```
1 2 0
0 0 ID #let ASReml count how many data records there are
Trait 0 US * #letting ASReml choose start values
```

2.5.1 Two rules for defining the residual error term

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

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

Rule 2 Where a compound model term is specified for the residuals, each combination of levels of the single model terms comprising this term must uniquely identify one unit of the data. For example, in the spatial analysis of a column trial comprising 4 replicates of 24 varieties arranged as a grid of 4 rows by 24 columns (rows are replicates), a first order separable autoregressive spatial variance structure for the residuals can be specified by the consolidated model term `ar1(column).ar1(row)`, where `column` and `row` are the appropriate

columns in the data file. However, the number of data units must be the product of the number of levels for `row` and the number of levels for `column`; 96 in this case. If this is not the case, or if more than one unit is associated with some row column combination, ASReml will return an error message and it will not be possible to use `ar1(column).ar1(row)` for residual error. If there are fewer than 96 units and each row-column combination present is associated with one unit, then Section 1.2.2 shows how to augment the data by completing the grid to allow an appropriate analysis.

These rules will always be satisfied for a single section of data, that is, $\boldsymbol{R}_v = \boldsymbol{R}_{v_1} = \sigma^2 \boldsymbol{I}_n$, see Example 2.2, defined either by default (ie. with no residual specified) or in terms of the units factor. However, a mismatch in both size and ordering is possible when either multiple sections are present (as in multi-environment trial (MET) analysis) or when non-identity variance model functions are used.

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

Section 2.3.5 described partitioning the data observations into data *sections* to which separate variance structures are applied. There are three data *sections* in the example *variance heterogeneity among the data observations* (*4.* in box on page 34). When variance structures are specified using dimensions rather than factor names (`idv(23)` for section 1, `idv(27)` for section 2,... in the example), the data must be ordered into sections and the variance structures must be ordered to match the order of the sections in the data file. It is usually more convenient to use a variable in the data file to identify sections within the data. The data will be sorted internally by ASReml (ie. the data file does not need to be ordered in any particular way) and the variance structures for sections can then be specified using the `sat` function, for example

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

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

- it tells ASReml that the variance structure for the residual error term is a direct sum structure (see Section 2.3.5) where the different components of the direct sum apply to the different levels of the sectioning variable in the data file

- it prunes the levels for a section so that *only the levels of factors defining the residual variance structure for that section are used in forming that variance structure.*

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

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

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

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

or, more succinctly, as

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

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

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

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

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

```
3 2 0
row row AR1 .1 #default estimate residual variance for all 3 sections
column column AR1 0.1
row row AR1 .1
column column ID
row row AR1 .1
column column AR1 0.1
```

2.6 Identifiability

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

model function in the consolidated model term includes variance parameters, for example `idv(A).ar1v(B)`, then the parameters will not all be identifiable and so the user must either change `idv(A)` to `id(A)` and leave `ar1v(B)` as it is, or change `ar1v(B)` to `ar1(B)` and leave `idv(A)` as it is.

2.7 A sequence of variance structures for the NIN data

Having outlined the theory and introduced the functional specification, we pause now to consider an example. The following is a series of six variance structures of increasing complexity for the NIN column trial data (see Chapter 3 of the User Guide for an introduction to these data). For each example we present a code box to the right that contains the functional specification, a discussion of this code to the left, and the matching code under the structural specification in a code box immediately below. We present the model specification explicitly to help the user understand the logic. In some cases, experienced users will wish to take advantage of reducing typing and clarity by using default rules. These are discussed in Section 2.11.

1 Randomised complete blocks analysis: blocks fixed

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

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

2.7 A sequence of variance structures for the NIN data

```
 NIN Alliance Trial 1989
 variety !A
 id
⋮
 row 22
 column 11
nin89.asd !skip 1
yield ~ mu variety repl
1 1 0
0 0 ID #default gamma parameterization
```

2 RCB analysis: blocks random

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

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

```
 NIN Alliance Trial 1989
 variety !A
 id
⋮
 row 22
 column 11
nin89.asd !skip 1
yield ~ mu variety,
!r repl
1 1 0
0 0 ID #default gamma parameterization
```

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

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

The NIN trial was actually laid out in as a rectangular array indexed in the data file by `row` and `column`. We can therefore consider fitting a *spatial* model for the residual term where we allow for autocorrelated errors in the row and/or column direction, see Section 2.5. However, there are missing plots in the original data. Before fitting a spatial analysis, we therefore need to fill out the data file to contain records for the missing plots. This allows us to define a separable variance structure for the residual error term that is the kronecker product of a structure for rows and a structure for columns. The example in the code box specifies $\boldsymbol{e} \sim N(\mathbf{0},\ \sigma^2_{e_c}\boldsymbol{I}_{11} \otimes \boldsymbol{\Sigma}_r(\rho_r))$, that is, a two-dimensional first order separable autoregressive spatial structure for error but with spatial correlation in the row direction only (IDV×AR1): `ar1(row)` models the $\boldsymbol{\Sigma}_r(\rho_r)$ correlation structure for rows and `idv(column)` models the IDV variance structure for columns. The consolidated model term

`idv(column).ar1(row)`

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

Important points

- the same residual variance structure could be achieved by specifying `id(column).ar1v(row)` which mirrors the alternate but equivalent algebraic form $\text{var}\,(\boldsymbol{e}) = \boldsymbol{I}_{11} \otimes \sigma^2_{e_r}\boldsymbol{\Sigma}_r(\rho_r)$. It is arbitrary which variable the common variance is attached to: `column` in the code box, `row` in the latter, see Section 2.6 on identifiability.

- if the correlation structure `id(column).ar1(row)` was specified, ASReml would automatically add a common variance to model $\text{var}\,(\boldsymbol{e}) = \sigma^2_e\boldsymbol{I}_{11} \otimes \boldsymbol{\Sigma}_r(\rho_r)$, see Section 2.6.

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

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

- this is a univariate analysis in which case ASReml automatically uses the gamma parameterization for estimation, see Section 2.8. Consequently, both the sigmas and the gammas are reported. The user can force ASReml to use the sigma parameterization by placing `!SIGMAP` immediately after the independent variable and before ∼ on the model definition line:

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

  `!SIGMAP` is a new qualifier with Release 4, see also Section 2.8. In this case only the sigmas are reported but they appear twice in the output, that is, in both of the columns headed `sigma` in the `.asr` file, see Chapter 11 of the User Guide for detailed information on output formats in ASReml.

```
NIN Alliance Trial 1989
 variety !A
 id
⋮
 row 22
 column 11
nin89aug.asd !skip 1
yield ~ mu variety !r repl !f mv
1 2 0
11 column ID #default gamma parameterization
22 row AR1
```

3b Two-dimensional separable autoregressive spatial model

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

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

Important points

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

- if the correlation structure `ar1(column).ar1(row)` was specified, ASReml would automatically add a common variance, see Section 2.6.

- ASReml would report an error if the consolidated model term `ar1v(column).ar1v(row)` was specified as this would correspond to $\text{var}(\boldsymbol{e}) = \sigma^2_{e_c}\boldsymbol{\Sigma}_c(\rho_c) \otimes \sigma^2_{e_r}\boldsymbol{\Sigma}_r(\rho_r)$ and $\sigma^2_{e_c}$ and $\sigma^2_{e_r}$ are unidentifiable, that is, it is not possible to estimate them separately, see Section 2.6.

```
NIN Alliance Trial 1989
 variety !A
 id
⋮
 row 22
 column 11
nin89aug.asd !skip 1
yield ~ mu variety !r repl !f mv
1 2 0
11 column AR1 #default gamma parameterization
22 row AR1
```

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

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

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

```
NIN Alliance Trial 1989
 variety !A
 id
⋮
 row 22
 column 11
nin89aug.asd !skip 1
yield ~ mu variety !r repl units !f mv
1 2 0
11 column AR1 #default gamma parameterization
22 row AR1
```

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

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

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

Important points

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

```
 NIN Alliance Trial 1989
 variety !A
 id
⋮
 row 22
 column 11
nin89aug.asd !skip 1
yield ~ mu variety !r repl mu.column.row !f mv
1 1 1
0 0 ID #default gamma parameterization
```

2.8 Sigma versus gamma parameterization

From Section 2.3.1, the variance matrix of $\boldsymbol{y}$ is

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

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

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

see (2.8). This is referred to as the *gamma parameterization* and the variance structure parameters in $\boldsymbol{\gamma}_g$ and $\boldsymbol{\gamma}_r$ are referred to as *gammas*, see Section 2.3.6.

2.8.1 Which parameterization does ASReml use for estimation?

ASReml switches between the sigma and the gamma parameterizations depending on the residual model specification. The current default for univariate, single section data-sets is the gamma parameterization. It is possible to over-ride this default as discussed in the following section. ASReml reports both the gammas and the sigmas when the gamma parameterization is used for estimation. For historical reasons, the sigmas are presented twice (two identical columns) when the sigma parameterization is used for estimation.

ASReml uses the sigma parameterization for analyses other than univariate 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).

2.8.2 Switching from the gamma to the sigma parameterization

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

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

would force ASReml to use the sigma parameterization in NIN example **3a,** see Section 2.7.

Table 2.3 gives the variance model specification for each of the six NIN examples (column 3), the individual terms in $\boldsymbol{G}(\boldsymbol{\sigma}_g)$ and $\boldsymbol{R}_v(\boldsymbol{\sigma}_r)$ under the sigma parameterization (column 4), the sigmas that are estimated under this parameterization (column 5), the individual terms in $\boldsymbol{G}(\boldsymbol{\gamma}_g)$ and $\boldsymbol{R}_v(\boldsymbol{\gamma}_r)$ under the gamma parameterization (column 6) and the gammas that are estimated under this parameterization (column 7).

Table 2.3: G structure for the random terms (magenta) and R structure for the residual error term (cyan) under both the sigma and gamma parameterizations, and the corresponding sigma(s)/gamma(s) under each parameterization for the series of NIN data examples

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

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

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

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

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

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

2.9 Variance model functions available in ASReml

The full range of variance models, that is, correlation, homogeneous variance and heterogeneous variance models available in ASReml is presented in Table 2.5 which is located at the end of this document for easy access, see Section 2.12 on page 54. This presents the variance structure name (in UPPERCASE), the corresponding variance model function name (in `lowercase`) used to associate the variance structure with the appropriate component of a model term, a brief description, the algebraic form of the model and the number of associated variance structure parameters.

The models span correlation (base) models (diagonal elements equal to 1 and correlations on the off diagonals), the extension of these to variance models (variances on the diagonals and covariance on the off diagonals), additional models that are parameterized as variance matrices rather than as correlation matrices and some special cases where the covariance structure is known except for the scale.

2.9.1 Forming variance models from correlation models

The variance function models presented under correlation models in Table 2.5 (`id` ... `mat`k) are used to specify the correlation models for the corresponding variance structures. The corresponding homogeneous and heterogeneous variance models are specified by appending `v` and `h` to the variance model function names respectively, and appending the corresponding variance parameters to the corresponding list of parameters. This convention holds for most models. It does not make sense to append `v` or `h` to the variance model function names for the heterogeneous variance models from `diag` ... `xfa`k.

In summary:

- to specify a correlation model, provide the variance model function name given in Table 2.5, for example, for a factor `row`

  `exp(row)`

  is an exponential correlation model with a single correlation parameter,

- to specify an homogeneous variance model, append a `v` to the variance model function name, for example

  `expv(row)`

  is an exponential variance model with 2 parameters (correlation and variance),

- to specify a heterogeneous variance model, append an `h` to the variance model function name, for example

  `coruh(site)`

  is a variance matrix with different variances for each site but the same correlation for all pairs of sites.

Important See Section 2.6 for rules on combining variance models and Section 2.10.1 for important notes regarding initial values.

2.10 New variance model function qualifiers

A consolidated model term is comprised of one or more covariance components, where a covariance component is a component of the model term to which a variance model function has been applied, see Section 2.4 and Table 2.1. All of the covariance components so far have been of the form

vmfname(component)

where *vmfname* is the variance model function name (in `this font` in first column of Table 2.5) and *component* is a component in the model term. Two single covariance components are `idv(repl)` and `ar1(row)`, see Table 2.1.

A general form for a covariance component is

vmfname(single_component qualifiers)

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

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

vmfname(ω qualifiers)

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

New qualifiers are in Table 2.4 (see Section 7.7 of the User Guide for the full list of qualifiers).

2.10 New variance model function qualifiers

In the structural specification the G structure is specified on *G structure lines* placed after any R *structure lines*. There must be a *variance header line* immediately after the model line. It specifies three numbers, s, c an g. s and c relate to the R structure and are defined on page 34. g is the number of (random) model terms which have a variance structure explicitly defined as described here.
Each specification begins with a *G structure header line*. It contains the name of the model term and the number of components. For each component, the variance structure is then specified, nominating the
size key structure initial-parameter-values qualifiers where
size is the name of the component or the number of effects in it
key is 0 for most G structure cases
structure is the name of variance structure
initial-parameter-values and *qualifiers* are discussed later.
In the structural specification it is not now mandatory to specify initial values.

Table 2.4: New variance model function qualifiers available in ASReml

qualifier	description
`!INIT` v	v is the list of initial values for the variance structure parameters. If. initial values can be obtained from the `.msv`, `.rsv` or `.tsv` file, they override these values, see Section 2.10.1
`!COORD` v	provides coordinates for mapping the effects so that a spatial model can be applied to the effects. It is needed when the coordinates are not in the data file, for example `exp(Trait !COORD 1 2.5 3.5 5 8)`, see Section 2.10.2.
`!SUBSECTION` f	f is a factor in the data that breaks the section into independent subsections, with subsections having common variance parameters, see Section 2.10.3
`!USE` t	t is a compound model term component used elsewhere in the model; allows this variance structure and its parameters to be the same as that used for t, see Section 2.10.4 for an example

For existing qualifiers, please refer to Chapter 7 in ASReml User Guide Release 4, Functional Specification.

2.10.1 Initial values `!INIT` v

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

There are several ways to provide initial values. The particular choice will depend on how

many values and other variance model function qualifiers are to be specified. The initial values can be provided in a number of ways:

- in the variance structure specification, for example

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

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

- by modifying the `.tsv` or `.msv` file created in a preliminary run (Section 1.1.2)

- by supplying an `.rsv` file using `!CONTINUE`, refer to the User Guide.

Important points

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

- for the gamma parameterization (Section 2.8), the variance structure parameters will be gammas; in this case the initial values for the gammas that are variance component ratios will be interpreted by ASReml as ratios.

The qualifiers are the same as for the structural specification except that:

- they are specified on the R and G structure lines in the structural specification, see (see pages 49 and 34)
- the `!INIT` qualifier name is not defined in the structural formulation because initial values are always expected
- the `!COORD` qualifier was not provided in the structural specification: if v was required, it was supplied as an unadorned vector after the G structure lines.

2.10.2 Ways to supply distances in one-dimensional metric based models !COORD v

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

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

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

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

where x is a vector of distances which has to be of length the number of levels of Time.

2.10.3 About subsections !SUBSECTION f

The !SUBSECTION qualifier provides an extension to the sat function of Section 2.5.2 for modelling the residual variance. It allows the case of modelling multiple independent sections of correlated observations with a common variance structure and common parameters within sections. The sections can be of different sizes and any homogeneous variance correlation model in Table 2.5 may be used for the variance structure. This gives an R structure of the form

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

so $\boldsymbol{R}_{v_i}$ may have a direct sum structure with common parameters. Note that, !SUBSECTION is only available when the residual variance function is expressed in terms of one variance function. !SUBSECTION f performs two tasks similar to those described in Section 2.5.2, that is, defining a direct sum structure for the residual vector in a section, with the number of subsections in section i, s_i, given by the number of levels of the factor f, and pruning the levels of the factor defining the variance structure within each section but allowing common variance parameters across sections. The data needs to be sorted in order of the variable f. The following code would specify a common AR1 structure across sections, assumed sorted in to the appropriate order within the section variable, with an initial spatial autocorrelation parameter of 0.5

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

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

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

to, specify a common EXP structure across plots.

2.10.4 Equating variance structures !USE t

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

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

we fit

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

requiring both xfa1 terms have the same parameters.

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

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

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

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

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

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

2.11 Default variance structures in ASReml

There are default variance structures in ASReml that allow the linear mixed model to be specified more succinctly. IDV is the default variance structure for random model terms and for the residual error terms. For example

- `A` will be interpreted as `idv(A)`
- `A.B` will be interpreted as `idv(A.B)`
- `A.B.C` will be interpreted as `idv(A.B.C)`
- `sat(Expt,1).A` will be interpreted as `sat(Expt,1).idv(A)`
- `sat(Expt,1).A.B` will be interpreted as `sat(Expt,1).idv(A.B)`
- `sat(Expt,1).A.B.C` will be interpreted as `sat(Expt,1).idv(A.B.C)`

In these cases the model term can be followed by an initial value and/or a parametric qualifier for example `A 1 !GP` is interpreted as `idv(A !INIT 1 !GP)` there is always a residual error term in the model but if it is not explicitly specified it is assumed to be `idv(units)`. If the consolidated model term definition is incomplete, that is, if some but not all of the

components have a variance model function specified, the variance model functions `idv()` or `id()` will be applied to these components depending on the variance model functions specified. For example

- `idv(A).B` will be interpreted as `idv(A).id(B)`
- `id(A).B` will be interpreted as `id(A).idv(B`
- `id(A).B.C` will be interpreted as `id(A).idv(B.C)`
- `idv(A).B.C` will be interpreted as `idv(A).id(B.C)`

Similarly, at the residual level as `sat()` cannot be converted into a variance function

- `sat(Expt,1).id(A).B` will be interpreted as `sat(Expt,1).id(A).idv(B)`
- `sat(Expt,1).id(A).B.C` will be interpreted as `sat(Expt,1).id(A).idv(B.C)`
- `sat(Expt,1).idv(A).B.C` will be interpreted as `sat(Expt,1).idv(A).id(B.C)`

However, it is good practice to specify variance model functions for the components in model terms and we encourage the user to do this. ASReml will automatically add a common variance to consolidated model terms that are specified as correlation models for both R and G structures, for example,

- `id(A)` will be converted to `idv(A)`
- `sat(Expt,1).id(units)` will be converted to `sat(Expt,1).idv(units)`
- `id(A).ar1(B)` will be converted to `idv(A).ar1(B)`
- `ar1(A).ar1(B)` will be converted to `ar1v(A).ar1(B)`
- `sat(Expt,1).id(A).ar1(B)` will be converted to `sat(Expt,1).idv(A).ar1(B)`
- `sat(Expt,1).ar1(A).ar1(B)` will be converted to `sat(Expt,1).ar1v(A).ar1(B)`

Using NIN example **2** for demonstration (Section 2.7), a more succinct coding of the model definition would be

```
yield ~ mu variety !r repl
residual units
```

which would result in identical output to the original example. The model could be relaxed further to

```
yield ~ mu variety !r repl
```

2.12 Variance models available in ASReml

Table 2.5: Details of the variance models available in ASReml

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

2.12 Variance models available in ASReml

Details of the variance models available in ASReml

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

2.12 Variance models available in ASReml

Details of the variance models available in ASReml

variance structure name	description	algebraic form	number of parameters†		
`variance model function name`			corr	hom variance	het variance
One-dimensional unequally spaced					
EXP `exp`	exponential	$C_{ii} = 1$ $C_{ij} = \phi^{\lvert x_i - x_j \rvert},\ i \neq j$ x_i are *coordinates* $0 < \phi < 1$	1	2	$1 + \omega$
GAU `gau`	gaussian	$C_{ii} = 1$ $C_{ij} = \phi^{(x_i - x_j)^2},\ i \neq j$ x_i are *coordinates* $0 < \phi < 1$	1	2	$1 + \omega$
Two-dimensional irregularly spaced					
		$\boldsymbol{x}$ and $\boldsymbol{y}$ vectors of coordinates $\theta_{ij} = \min(d_{ij}/\phi_1, 1)$ d_{ij} is euclidean distance			
IEXP `iexp`	isotropic exponential	$C_{ii} = 1$ $C_{ij} = \phi^{\lvert x_i - x_j \rvert + \lvert y_i - y_j \rvert},\ i \neq j$ $0 < \phi < 1$	1	2	$1 + \omega$
IGAU `igau`	isotropic gaussian	$C_{ii} = 1$ $C_{ij} = \phi^{(x_i - x_j)^2 + (y_i - y_j)^2},\ i \neq j$ $0 < \phi < 1$	1	2	$1 + \omega$
IEUC `ieuc`	isotropic euclidean	$C_{ii} = 1$ $C_{ij} = \phi^{\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}},\ i \neq j$ $0 < \phi < 1$	1	2	$1 + \omega$
LVR `lvr`	linear variance	$C_{ij} = (1 - \theta_{ij})$ $0 < \phi_1$	1	2	$1 + \omega$

2.12 Variance models available in ASReml

Details of the variance models available in ASReml

variance structure name `variance model function name`	description	algebraic form	number of parameters† corr	hom variance	het variance
SPH `sph`	spherical	$C_{ij} = 1 - \frac{3}{2}\theta_{ij} + \frac{1}{2}\theta_{ij}^3$ $0 < \phi_1$	1	2	$1 + \omega$
CIR `cir`	circular (Webster & Oliver, 2001, p 113)	$C_{ij} = 1$ $- \frac{2}{\pi}(\theta_{ij}\sqrt{1-\theta_{ij}^2} + \sin^{-1}\theta_{ij})$ $0 < \phi_1$	1	2	$1 + \omega$
AEXP `aexp`	anisotropic exponential	$C_{ii} = 1$ $C_{ij} = \phi_1^{\lvert x_i - x_j\rvert}\phi_2^{\lvert y_i - y_j\rvert}$ $0 < \phi_1 < 1$, $0 < \phi_2 < 1$	2	3	$2+\omega$
AGAU `agau`	anisotropic gaussian	$C_{ii} = 1$ $C_{ij} = \phi_1^{(x_i - x_j)^2}\phi_2^{(y_i - y_j)^2}$ $0 < \phi_1 < 1$, $0 < \phi_2 < 1$	2	3	$2 + \omega$
MATk `mat`k	Matérn with first $1 \le k \le 5$ parameters spec-ified by the user	C_{ij} =Matérn: see text $\phi > 0$ range, ν shape(0.5) $\delta > 0$ anisotropy ratio(1), α anisotropy angle(0), $\lambda(1\vert 2)$ metric(2)	k	k+1	$k + \omega$
heterogeneous variance models					
DIAG `diag`	diagonal = `IDH` `idh`	$\boldsymbol{\Sigma}_{ii} = \phi_i$ $\boldsymbol{\Sigma}_{ij} = 0$, $i \ne j$	-	-	ω
US us	unstructured general covariance matrix	$\boldsymbol{\Sigma}ij = \phi_{ij}$	-	-	$\frac{\omega(\omega+1)}{2}$
OWNk `own`k	user explicitly forms $\boldsymbol{V}$ and $\partial\boldsymbol{V}$		-	-	k

2.12 Variance models available in ASReml

Details of the variance models available in ASReml

variance structure name `variance model function name`	description	algebraic form	number of parameters† corr	hom variance	het variance
ANTE1 `ante1` ANTEk `ante`k	1^{st} k^{th} k order antedependence $1 \leq k \leq \omega - 1$	$\boldsymbol{\Sigma}^{-1} = \boldsymbol{U}\boldsymbol{D}\boldsymbol{U}^{\mathsf{T}}$ $D_{ii} = d_i,\ D_{ij} = 0,\ i \neq j$ $U_{ii} = 1,\ U_{ij} = u_{ij},\ 1 \leq j - i \leq k$ $\boldsymbol{U}_{ij} = 0,\ i > j$	-	-	$\frac{\omega(\omega+1)}{2}$
CHOL1 `chol1` CHOLk `chol`k	1^{st} k^{th} k order cholesky $1 \leq k \leq \omega - 1$	$\boldsymbol{\Sigma} = L\boldsymbol{D}L^{\mathsf{T}}$ $D_{ii} = d_i,\ D_{ij} = 0,\ i \neq j$ $L_{ii} = 1,\ L_{ij} = l_{ij},\ 1 \leq i - j \leq k$	-	-	$\frac{\omega(\omega+1)}{2}$
FA1 `fa1` FAk `fa`k	1^{st} k^{th} k order factor analytic	$\boldsymbol{\Sigma} = \boldsymbol{D}\boldsymbol{C}\boldsymbol{D}$, $\boldsymbol{C} = \boldsymbol{F}\boldsymbol{F}^{\mathsf{T}} + \boldsymbol{E}$, $\boldsymbol{F}$ contains k correlation factors $\boldsymbol{E}$ diagonal $\boldsymbol{D}\boldsymbol{D} = \mathrm{diag}\,(\boldsymbol{\Sigma})$	-	-	$\omega + \omega$ $k\omega + \omega$
FACV[1] `facv1` FACVk `facv`k	1^{st} k^{th} k order factor analytic covariance form	$\boldsymbol{\Sigma} = \boldsymbol{\Gamma}\boldsymbol{\Gamma}^{\mathsf{T}} + \boldsymbol{\Psi}$, $\boldsymbol{\Gamma}$ contains covariance factors $\boldsymbol{\Psi}$ contains specific variance	-	-	$\omega + \omega$ $k\omega + \omega$
XFA1 `xfa1` XFAk `xfa`k	1^{st} k^{th} k order extended factor analytic	$\boldsymbol{\Sigma} = \boldsymbol{\Gamma}\boldsymbol{\Gamma}^{\mathsf{T}} + \boldsymbol{\Psi}$, $\boldsymbol{\Gamma}$ contains covariance factors $\boldsymbol{\Psi}$ contains specific variance	-	-	$\omega + \omega$ $k\omega + \omega$

Details of the variance models available in ASReml

variance structure name	description	algebraic form	number of parameters†		
variance model function name			corr	hom variance	het variance
relationship matrices‡					
NRM nrm	relationship matrix derived from pedigree		0	1	-
GRM1 grm1	generalized relationship number 1		0	1	-
⋮	⋮	⋮	⋮	⋮	
GRM8 grm8	generalized relationship matrix 8		0	1	-

† This is the number of variance structure parameters, ω is the dimension of the matrix. The homogeneous variance form is specified by appending `V` to the correlation basename; the heterogeneous variance form is specified by appending `H` to the correlation basename

‡ These will be associated with 1 variance parameter unless used in direct product with another structure that provides the variance. Appending a `v` to a name makes it explicit that a variance parameter is fitted.

2.13 Functions of variance components using names

Rather than identifying parameters by number when specifying functions in `VPREDICT ASReml4` allows parameters to be named. When ASReml reads back the variance parameters from the `.asr` file, each covariance component, or variance function, is assigned a name. The functional specification allows a more explicit naming convention than the structural specification. The full name is usually the covariance function, or its specified contracted form, prepended by the consolidated model term, or its specified contracted form, and the symbol ;. Exceptions to this rule are single components `F`, `id[v](F)` and `nrm[v](F)` terms which are reduced to the corresponding single term `F`, `id[v](F)` and `nrm[v](F)`. So, for example, with the random model and residual specification model terms

```
!r idv(A) ar1v(B) nrm(C).us(Trait) D residual id(units).us(Trait)
```

The covariance functions with parameters

`idv(A),ar1v(B), us(Trait)` in `nrm(C).us(Trait)`

and

`us(Trait)` in `id(units).us(Trait)` are named

`idv(A)`, `ar1v(B);ar1v(B)`, `nrm(C).us(Trait);us(Trait)`, `id(units).us(Trait);us(Trait)`. If the resulting name is not ambiguous the name can be contracted by reducing the consolidated model term to a unique substring or leaving out the consolidated model term completely. For example, in the example the covariance functions can be represented by `idv(A)`, `ar1v(B)`, `C;us(Trait)` and `units;us(Trait)`, respectively. Individual parameters within a covariance component can be specified by number , or sequence of numbers (`n:m`) by appending these in square braces, for example, `C;us(Trait)[3]` or `units;us(Trait)[4:6]`. If the residual directive is not used, the default R structure parameters are effectively named `Residual`.

2.13.1 A detailed example

The following example for a **bivariate sire model** is a little more complicated. The job file `bsiremod.as` contains

```
...
coop.fmt

ywt fat ~ Trait Trait.(age c(brr) sex sex.age) !r us(Trait).id(sire);us(Trait)  !f Tr.grp
residual id(units).us(Trait)

VPREDICT !DEFINE
F phenvar id(units).us(Trait);us(Trait) + us(Trait).sire;us(Trait) # 1:3 + 4:6
F addvar sire;us(Trait) * 4                                       # 4:6 * 4
H heritA addvar[1] phenvar[1]                                     # 10 7
H heritB addvar[3] phenvar[3]                                     # 12 9
R phencorr phenvar                                                # 7 8 9
R gencorr addvar                                                  # 4:6
```

The relevant lines of the `.asr` file are

```
 Model_Term                          Sigma         Sigma   Sigma/SE   % C
 id(units).us(Trait)          8140 effects
 Trait                  US_V  1  1   23.2055       23.2055      44.44   0 P
 Trait                  US_C  2  1   2.50402       2.50402      18.56   0 P
 Trait                  US_V  2  2   1.66292       1.66292      32.82   0 P
 us(Trait).id(sire)            184 effects
 Trait                  US_V  1  1   1.45821       1.45821       3.66   0 P
 Trait                  US_C  2  1  0.130280      0.130280       1.92   0 P
 Trait                  US_V  2  2  0.344381E-01  0.344381E-01   2.03   0 P
```

Numbering the parameters reported in `bsiremod.asr` (and `bsiremod.vvp`)

1 error variance for `ywt`
2 error covariance for `ywt` and `fat`
3 error variance for `fat`
4 sire variance component for `ywt`
5 sire covariance for `ywt` and `fat`
6 sire variance for `fat`

then

`F phenvar id(units).us(Trait);us(Trait) + us(Trait).id(sire);us(Trait)` or

2.13 Functions of variance components using names

`F phenvar units;us(Trait) + sire;us(Trait)` or `F phenvar 1:3 + 4:6`

creates new components **7** = **1**+**4**, **8** = **2**+**5** and **9** = **3**+**6**,

`F addvar sire;us(Trait) * 4` or `F addvar 4:6 * 4`

creates new components **10** = **4** × 4, **11** = **5** × 4 and **12** = **6** × 4,

`H heritA addvar[1] phenvar[1]` or `H heritA 10 7`

forms **10** / **7** to give the heritability for `ywt`,

`H heritB addvar[3] phenvar[3]` or `H heritB 12 9`

forms **12** / **9** to give the heritability for `fat`,

`R phencorr phenvar` or `R phencorr 7 8 9`

forms **8** $/\sqrt{\mathbf{7} \times \mathbf{9}}$, that is, the phenotypic correlation between `ywt` and `fat`,

`R gencorr addvar` or `R gencorr 4:6`

forms **5** $/\sqrt{\mathbf{4} \times \mathbf{6}}$, that is, the genetic correlation between `ywt` and `fat`.

The resulting `.pvc` file contains:

```
id(units).us(Trait)           8140 effects
  1 id(units).us(Trait);us(Trait)           V  1  1      23.2055        0.522176
  2 id(units).us(Trait);us(Trait)           C  2  1      2.50402        0.134915
  3 id(units).us(Trait);us(Trait)           V  2  2      1.66292        0.506679E-01
us(Trait).id(sire)             184 effects
  4 us(Trait).id(sire);us(Trait)            V  1  1      1.45821        0.398418
  5 us(Trait).id(sire);us(Trait)            C  2  1     0.130280        0.678542E-01
  6 us(Trait).id(sire);us(Trait)            V  2  2     0.344381E-01    0.169646E-01
  7 phenvar  1                24.664       0.64250
  8 phenvar  2                2.6343       0.14763
  9 phenvar  3                1.6974       0.52365E-01
 10 addvar  4                 5.8328        1.5926
 11 addvar  5                0.52112       0.27168
 12 addvar  6                0.13775       0.67791E-01
    heritA       = addvar    10/phenvar    7=          0.2365    0.0612
    heritB       = addvar    12/phenvar    9=          0.0812    0.0394
    phenco  2  1 = phenv  8/SQR[phenv  7*phenv  9]=    0.4071    0.0183
    gencor  2  1 = addva 11/SQR[addva 10*addva 12]=    0.5814    0.2039
Notice: The parameter estimates are followed by
        their approximate standard errors.
```

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

3 Examples

This chapter takes examples from the User Guide and compares the functional specification to the structural specification. Please refer to the User Guide for a comprehensive description of each dataset and analysis.

Throughout this chapter, `this font and colour` are used for code that is common to both specifications. The first three examples (3.1 to 3.3) involve uncorrelated G and R structures, that is, the IDV variance structure for both the random model terms and the residual error term. As such, the explicit functional specification can be simplified to the structural specification. In these examples, `this font and colour` are used for `STRUCTURAL AND IMPLICIT FUNCTIONAL` code and `this font and colour` are used for `EXPLICIT FUNCTIONAL` code. In the remaining examples (from 3.4), `this font and colour` are used for `STRUCTURAL` code and `this font and colour` are used for `FUNCTIONAL` code. Note that, these examples present two alternative model specifications. To run the jobs retaining both model specifications in the command file, one of the specifications could be commented out.

3.1 Split plot design - Oats

```
split plot example
 blocks 6 #coded 1...6 in first data field of oats.asd
 nitrogen !A 4 #coded alphabetically
 subplots * #coded 1...4
 variety !A 3 #coded alphabetically
 wplots * #coded 1...3
 yield
oats.asd !SKIP 2
#STRUCTURAL and IMPLICIT FUNCTIONAL
yield ~ mu variety nitrogen variety.nitrogen !r blocks blocks.wplots
#EXPLICIT FUNCTIONAL
yield ~ mu variety nitrogen variety.nitrogen !r idv( blocks) idv(blocks.wplots)
residual idv(units)
```

3.2 Unbalanced nested design - Rats

```
Rats example
 dose 3 !A
 sex 2 !A
 littersize
 dam 27
 pup 18
 weight
rats.asd !DOPATH 1 #change DOPATH argument to select each PATH
!PATH 1
#STRUCTURAL AND IMPLICIT FUNCTIONAL
weight ~ mu littersize dose sex dose.sex !r dam
#EXPLICIT FUNCTIONAL
weight ~ mu littersize dose sex dose.sex !r idv(dam)
residual idv(units)
!PATH 2
#STRUCTURAL AND IMPLICIT FUNCTIONAL
weight ~ mu out(66) littersize dose sex dose.sex !r dam
#EXPLICIT FUNCTIONAL
weight ~ mu out(66) littersize dose sex dose.sex !r idv(dam)
residual idv(units)
!PATH 3
#STRUCTURAL AND IMPLICIT FUNCTIONAL
weight ~ mu littersize dose sex !r dam
#EXPLICIT FUNCTIONAL
weight ~ mu littersize dose sex !r idv(dam)
residual idv(units)
!PATH 4
#STRUCTURAL AND IMPLICIT FUNCTIONAL
Weight ~ mu littersize dose sex
#EXPLICIT FUNCTIONAL
weight ~ mu littersize dose sex
residual idv(units)
```

3.3 Source of variability in unbalanced data - Volts

```
Voltage data
 teststat 4 #4 testing stations tested each regulator
 setstat !A #10 setting stations each set 4-8 regulators
 regulator 8 #regulators numbered within setting stations
 voltage
voltage.asd !skip 1
#STRUCTURAL AND IMPLICIT FUNCTIONAL
voltage ~ mu !r setstat setstat.regulator teststat setstat.teststat
```

```
#EXPLICIT FUNCTIONAL
voltage ~ mu !r idv(setstat) idv(setstat.regulator) idv(teststat),
idv(setstat.teststat)
residual idv(units)
```

The next three examples gives examples of different residual structures.

3.4 Balanced repeated measures - Height

```
This is plant data multivariate
 tmt !A #diseased Healthy
 plant 14
 y1 y3 y5 y7 y10
grass.asd !skip 1 !ASUV
!Y y1 !G tmt !JOIN #plot the data
```

First a split plot in time model can be fitted by fitting a units term plus an independent residual.

```
#STRUCTURAL
y1 y3 y5 y7 y10 ~ Trait tmt Trait.tmt !r units
1 2 0 #1 section 2 dimensions 0 G structure
14 0 ID #14 units no reordering ID structure can reduce to 14
Trait 0 ID
#FUNCTIONAL
y1 y3 y5 y7 y10 ~ Trait tmt Trait.tmt !r idv(units)
residual idv(units.Trait)
```

Next a split plot in time model can be fitted by specifying a CORU variance model for the R structure

```
#STRUCTURAL
y1 y3 y5 y7 y10 ~ Trait tmt Trait.tmt
1 2 0
14
Trait 0 CORU .5
#FUNCTIONAL
y1 y3 y5 y7 y10 ~ Trait tmt Trait.tmt
residual idv(units).coru(Trait)
```

Next an exponential (EXP) model to allow the correlation between observations to decline with time differences.

```
#STRUCTURAL
y1 y3 y5 y7 y10 ~ Trait tmt Trait.tmt
1 2 0
14
Trait 0 EXP .5
```

```
1 3 5 7 10 #time coordinates
#FUNCTIONAL
y1 y3 y5 y7 y10 ~ Trait tmt Trait.tmt
residual id(units).exp(Trait !INIT 0.5 !COORD 1 3 5 7 10)
```

Next an exponential model with heterogeneous variances (EXPH) is fitted

```
#STRUCTURAL
y1 y3 y5 y7 y10 ~ Trait tmt Trait.tmt
1 2 0
14 !S2==1
Trait 0 EXPH .5 100 200 300 300 300
1 3 5 7 10
#FUNCTIONAL
y1 y3 y5 y7 y10 ~ Trait tmt Trait.tmt
residual id(units).exph(Trait !INIT 0.5 100 200 300 300 300 !COORD 1 3 5 7 10 )
```

Next an antedependence model with heterogeneous variances (EXPH) is fitted

```
!ASSIGN ANTE1 !< !INIT 60.16
54.65 73.65
91.50 123.3 306.4
89.17 120.2 298.6 431.8
62.21 83.85 208.3 301.2 379.8 !>
#ASSIGN used to make input more legible
residual units.ante(Trait !INIT $ANTE1)
#STRUCTURAL
y1 y3 y5 y7 y10 ~ Trait tmt Trait.tmt
1 2 0
14 !S2==1
Trait 0 ANTE $ANTE1
#FUNCTIONAL
y1 y3 y5 y7 y10 ~ Trait tmt Trait.tmt
residual idv(units).ante(Trait !INIT $ANTE1)
```

Finally, an unstructured (US) matrix is fitted.

```
!ASSIGN US1 !< !INIT 37.20
23.38 41.55
34.83 61.89 258.9
44.58 79.22 331.4 550.8
43.14 76.67 320.7 533.0 541.4 !>
#STRUCTURAL
y1 y3 y5 y7 y10 ~ Trait tmt Trait.tmt
1 2 0
14 !S2==1
Trait 0 US $US1
#FUNCTIONAL
y1 y3 y5 y7 y10 ~ Trait tmt Trait.tmt
```

```
residual id(units).us(Trait !INIT $US1)
```

An alternative way of fitting an unstructured error model for multivariate data is to omit the `!ASUV` qualifier.

```
#STRUCTURAL
y1 y3 y5 y7 y10 ~ Trait tmt Trait.tmt
1 2 0
0
Trait 0 US * #ASReml generates initial values
#FUNCTIONAL
y1 y3 y5 y7 y10 ~ Trait tmt Trait.tmt
residual id(units).us(Trait) #ASReml generates initial values
```

3.5 Spatial analysis of a field experiment - Barley

```
!EPS !RENAME !ARG 1 2 3
Slate Hall example
 Rep 6 #six replicates of 5x5 plots in 2x3 arrangement
 RowBlk 30 #rows within replicates numbered across replicates
 ColBlk 30 #columns within replicates numbered across replicates
 row 10 #field row
 column 15 #field column
 variety 25
 yield
barley.asd !skip 1 !DOPATH $1
!PATH 1 #AR1 x AR1
#STRUCTURAL
y ~ mu var
1 2
15 column AR1 0.1 #second field is specified so ASReml can sort
10 row AR1 0.1 #records properly into field order
#FUNCTIONAL
y ~ mu var
residual ar1v(column).ar1(row)
!PATH 2 #AR1 x AR1 + units
#STRUCTURAL
y ~ mu var !r units
1 2
15 column AR1 0.1
10 row AR1 0.1
#FUNCTIONAL
y ~ mu var !r idv(units)
residual ar1v(column).ar1(row)
!PATH 3 #incomplete blocks
#STRUCTURAL
y ~ mu var !r Rep Rowblk Colblk
```

```
#FUNCTIONAL
y ~ mu var !r idv(Rep) idv(Rowblk) idv(Colblk)
residual idv(units)
```

3.6 Unreplicated early generation variety trial - Wheat

```
!EPS !RENAME !ARG 1 2 3 4
Tullibigeal Trial !DOPART $1
 linenum
 yield
 weed
 column 10
 row 67
 variety 532 #testlines 1:525, check lines 526:532
wheat.asd !SKIP 1
#AR1 x ID
!PART 1
#STRUCTURAL
y ~ mu weed mv !r variety
1 2 0
67 row AR1 0.1
10 column ID
#FUNCTIONAL
y ~ mu weed mv !r idv(variety)
residual ar1v(row).id(col)
#AR1 x AR1
!PATH 2
#STRUCTURAL
y ~ mu weed mv !r variety
1 2 0
67 row AR1 0.1
10 column AR1 0.1
#FUNCTIONAL
y ~ mu weed mv !r variety
residual ar1v(row).ar1(col)
!PATH 3 #AR1 x AR1 + column trend
#STRUCTURAL
y ~ mu weed pol(column,-1) mv !r variety
1 2 0
67 row AR1 0.1
10 column AR1 0.1
#FUNCTIONAL
y ~ mu weed pol(column,-1) mv !r variety
residual ar1v(row).ar1(col)
!PATH 4 #AR1 x AR1 + nugget + column trend
#STRUCTURAL
```

```
y ~ mu weed pol(column,-1) mv !r variety units
1 2 0
67 row AR1 0.1
10 column AR1 0.1
#FUNCTIONAL
y ~ mu weed pol(column,-1) mv !r variety units
residual ar1v(row).ar1(col)
```

3.7 Paired Case-Control study - Rice

This includes 3 analyses, one using uncorrelated effects, the second allowing heterogeneous variances to some effects, and thirdly a bivariate analysis. The data for the first two analyses is in file `rice.asd` and the data for the third analysis is in `ricem.asd`. The input for the first two analyses is:

```
Bloodworm data Dr M Stevens
 pair 132
 rootwt
 run 66
 tmt 2 !A
 id
 variety 44 !A
rice.asd !skip 1
!DOPATH 1 #change to 2 for second analysis
!PATH 1
#STRUCTURAL AND IMPLICIT FUNCTIONAL
sqrt(rootwt) ~ mu tmt !r variety variety.tmt run pair run.tmt
#EXPLICIT FUNCTIONAL
sqrt(rootwt) ~ mu tmt !r idv(variety) idv(variety.tmt) idv(run),
idv(pair) idv(run.tmt)
residual idv(units)
!PATH 2
#STRUCTURAL
sqrt(rootwt) ~ mu tmt !r variety tmt.variety run pair,
tmt.run uni(tmt,2)
0 0 2
tmt.variety 2
2 0 DIAG .1 .1 !GU
44 0 0
tmt.run 2
2 0 DIAG .1 .1 !GU
66 0 0
#FUNCTIONAL
sqrt(rootwt) ~ mu tmt !r idv(variety) diag(tmt).idv(variety) idv(run) idv(pair),
diag(tmt).id(run) uni(tmt,2)
residual idv(units)
```

The input for the third analysis is

```
  id
  pair 132
  run 66
  variety 44 !A
  yc #Control tray
  ye #Treated tray
ricem.asd !skip 1
!X yc !Y ye
tabulate sqrt(yc) sqrt(ye) ~ var run !COUNT
#STRUCTURAL
sqrt(yc) sqrt(ye) ~ Trait !r Trait.variety Trait.run
1 2 2
132 !S2==1
Trait 0 US 2.21 1.1 2.427
Trait.variety 2
2 0 US 1.401 1 1.477
44 0 0
Trait.run 2
2 0 US .79 .5 2.887
66 0 0
PREDICT variety
#FUNCTIONAL
sqrt(yc) sqrt(ye) ~ Trait !r us(Trait).id(variety) us(Trait).id(run)
residual id(units).us(Trait)
PREDICT variety
```

3.8 Balanced longitudinal data - Random coefficients and cubic smoothing splines - Oranges

This includes initial analyses on one tree,random coefficient effect and smoothing splines models

```
!RENAME !ARG 1 #First analysis:  replace with 2 for next analysis
!DOPART $1
this is the orange data,
seq #record number is not used
tree 5
age #118 484 664 1004 1231 1372 1582
circ
season !L Spring Autumn
!PART 1 #INDIVIDUAL TREE
orange.asd !skip 1 !filter tree !select 1
!PART 2
orange.asd !skip 1
!PART 0
```

```
!SPLINE spl(age,7) 118 484 664 1004 1231 1372 1582
!PVAL age 150 200:1500
!PATH 1
#STRUCTURAL AND IMPLICIT FUNCTIONAL
circ ~ mu age !r spl(age,7)
#EXPLICIT FUNCTIONAL
circ ~ mu age !r idv(spl(age,7))
residual idv(units)
!PATH 2
#STRUCTURAL
circ ~ mu age !r !{tree 4.6 age.tree .000094!} spl(age,7) .1,
spl(age,7).tree 2.3 fac(age) 13.9
0 0 1
tree 2
2 0 US 4.6 .00001 .000094
5 0 0
#FUNCTIONAL
circ ~ mu age !r str(tree age.tree us(2).tree !INIT 4.6 .00001 .000094),
idv(spl(age,7) !INIT .1) idv(spl(age,7).tree !INIT 2.3),
idv(fac(age) !INIT 13.9)
residual idv(units)
```

3.9 Generalised linear (mixed) models

3.9.1 Binomial analysis of Footrot score

```
Lamb data from ARG thesis page 177-8
 Year GRP 5 !V99=V2 !==4 !M1
 SEX SIRE !I
 Total
 FS1 FS2 Scald !+V99 Rot !+V99
 pRot !=Rot !/Total
#1 1 1 101 39 33 6 6 1
LAMB.DAT !skip 1
!DF 1904 !YSS 62.54249
#STRUCTURAL AND IMPLICIT FUNCTIONAL
pRot !TOTAL=Total ~ mu SEX GRP !r SIRE
#EXPLICIT FUNCTIONAL
pRot !TOTAL=Total ~ mu SEX GRP !r idv(SIRE)
residual idv(units)
```

An analysis of footrot as a binomial variable using the logistic link is performed by the model line (and dropping the !DF qualifier)

```
#STRUCTURAL AND IMPLICIT FUNCTIONAL
Rot !BIN !TOTAL=Total  mu SEX GRP SEX.GRP !r SIRE .16783
```

```
#EXPLICIT FUNCTIONAL
Rot !BIN !TOTAL=Total  mu SEX GRP SEX.GRP !r idv(SIRE !INIT .16783)
```

3.9.2 Bivariate analysis of Footrot score

```
Lamb data from ARG thesis page 177-8
 Year GRP 5 !V99=V2 !==4 !M 1
 SEX SIRE !I
 Score1
 Score2 Scald !+V99 Rot !+V99
 YVar
binnor.txt !skip 1 !ASUV !MAXIT 40
#STRUCTURAL
Score1 YVar !bin ~ Trait.SEX Trait.GRP !r Trait.SIRE
1 2 1
2513
2 0 US !GFPP
1 .01 0.25
Trait.SIRE 2
Trait 0 US 0.015 0.01 1.05
SIRE 0 ID
#FUNCTIONAL
Score1 YVar !bin ~ Trait.SEX Trait.GRP !r us(Trait).id(SIRE)
residual id(units).us(Trait !INIT 1 -.01 .25 !GFPP)
```

3.10 Multivariate animal genetics data - Sheep

```
!RENAME 1 !ARG 1 2 3 4 5 #Does 5 runs one for each trait
Multivariate Sire & Dam model
!DOPATH $1
!IF $1 == 1 !ASSIGN YV wwt #sets up dependent variable to each trait in turn
!IF $1 == 2 !ASSIGN YV ywt
!IF $1 == 3 !ASSIGN YV gfw
!IF $1 == 4 !ASSIGN YV fdm
!IF $1 == 5 !ASSIGN YV fat
tag
sire 92 !I
dam 3561 !I
grp 49
sex
brr 4
litter 4871
age
wwt !M0 # !M0 identifies missing values
ywt !M0
gfw !M0
```

3.10 Multivariate animal genetics data - Sheep

```
fdm !MO
fat !MO
coop.fmt
#STRUCTURAL
!PATH 1 2 3 5
$YV   mu age brr sex age.sex !r sire dam lit age.grp sex.grp !f grp #traits are substituted
for $YV
!PATH 4 #leaves out sex.grp for fdm
$YV   mu age brr sex age.sex !r idv(sire) idv(dam) idv(lit) idv(age.grp) !f grp #$fdm is substit
#FUNCTIONAL
!PATH 1 2 3 5
$YV   mu age brr sex age.sex !r idv(sire) idv(dam) idv(lit) idv(age.grp) idv(sex.grp) !f grp
#traits are substituted for$YV
!PATH 4 #leaves out sex.grp for fdm
$YV   mu age brr sex age.sex !r idv(sire) idv(dam) idv(lit) idv(age.grp) !f grp #$fdm is
#substituted for $YV


!RENAME 1 !ARG 3 #CHANGE 1 TO 2 OR 3 FOR OTHER PATHS
Multivariate Sire & Dam
!DOPATH $1
tag
sire 92 !I
dam 3561 !I
grp 49
sex
brr 4
litter 4871
age
wwt !MO # !MO identifies missing values
ywt !MO
gfw !MO
fdm !MO
fat !MO
!PATH 1 // coop.fmt
!PATH 2 // coop.fmt !CONTINUE
!PATH 3 // coop.fmt !CONTINUE

!PATH 0 #USING SUBSET TO SET UP COMBINATIONS OF TRAITS USED IN MODEL
!SUBSET TrDam123 Trait 1 2 3 0 0
!SUBSET TrLit1234 Trait 1 2 3 4 0
!SUBSET TrAG1245 Trait 1 2 4 5
!SUBSET TrSG123 Trait 1 2 3 0 0
!SUBSET TrDam12 Trait 1 2 0 0 0
!SUBSET TrDa12 Trait 1 2 0 0 0

#USING !ASSIGN TO MAKE SPECIFICATION CLEARER
#STRUCTURAL
#ASSIGN SIRE DAM LITTER AND RESIDUAL INITIAL VALUES FROM UNIVARIATE ANALYSES
!ASSIGN SDIAGI 0.608 1.298 0.015 0.197 0.035 #Initial sire variances
!ASSIGN DDIAGI 2.2 4.14 0.018
!ASSIGN LDIAGI 3.74 0.97 0.019 0.941
```

3.10 Multivariate animal genetics data - Sheep

```
!ASSIGN RUSI !< 9.27 0.0 16.48 0.0 0.0 0.14
0.0 0.0 0.0 3.37 0.0 0.0 0.0 0.0 1.14 !>
!ASSIGN VARS !<
TrAG1245.age.grp,
TrSG123.sex.grp !>

#FUNCTIONAL
#ASSIGN SIRE DAM LITTER AND RESIDUAL INITIAL VALUES FROM UNIVARIATE ANALYSES
!ASSIGN SDIAGI !INIT 0.608 1.298 0.015 0.197 0.035 #Initial sire variances
!ASSIGN DDIAGI !INIT 2.2 4.14 0.018
!ASSIGN LDIAGI !INIT 3.74 0.97 0.019 0.941
!ASSIGN RUSI !< !INIT 9.27 0.0 16.48 0.0 0.0 0.14
0.0 0.0 0.0 3.37 0.0 0.0 0.0 0.0 1.14 !>
!ASSIGN VARF !<
diag(TrAG1245 !INIT 0.0024 0.0019 0.0020 0.00026).age.grp,
diag(TrSG123 !INIT 0.93 16.0 0.28).sex.grp !>


!PATH 1 #DIAGONAL FOR SIRE DAM AND LITTER UNSTRUCTURED FOR RESIDUAL
#STRUCTURAL
wwt ywt gfw fdm fat   Trait Trait.age Trait.brr Trait.sex Trait.age.sex !r $VARS,
Trait.sire TrDam123.dam TrLit1234.lit ,
!f Trait.grp
1 2 5 #1 R STRUCTURE WITH 2 COMPONENTS AND 5 G STRUCTURES
0 0 0 #INDEPENDENT ACROSS ANIMALS
Trait 0 US !GP #UNSTRUCTURED TRAIT MATRIX INITIAL VALUES FROM UNIVARIATE ANALYSES
$RUSI
TrAG1245.age.grp 2
TrAG1245 0 DIAG 0.0024 0.0019 0.0020 0.00026
age.grp 0 ID
TrSG123.sex.grp 2
TrSG123 0 DIAG 0.93 16.0 0.28
sex.grp 0 ID
Trait.sire 2
Trait 0 DIAG $SDIAGI
sire 0 ID
TrDam123.dam 2
TrDam123 0 DIAG $DDIAGI
dam 0 ID
TrLit1234.lit 2
TrLit1234 0 DIAG $LDIAGI
lit 0 ID


#FUNCTIONAL
wwt ywt gfw fdm fat   Trait Trait.age Trait.brr Trait.sex Trait.age.sex !r $VARF,
diag(Trait $SDIAGI).sire diag(TrDam123 $DDIAGI).dam diag(TrLit1234 $LDIAGI ).lit,
!f Trait.grp
residual id(units).us(Trait $RUSI)



!PATH 2 #CHANGE DIAGONAL TO XFA1 FOR SIRE DAM AND LITTER
#STRUCTURAL
```

3.10 Multivariate animal genetics data - Sheep

```
wwt ywt gfw fdm fat   Trait Trait.age Trait.brr Trait.sex Trait.age.sex !r $VARS,
xfa(Trait,1).sire xfa(TrDam123,1).dam xfa(TrLit1234,1).lit ,
!f Trait.grp mv
1 2 5 #1 R STRUCTURE WITH 2 COMPONENTS AND 5 G STRUCTURES
0 0 0 #INDEPENDENT ACROSS ANIMALS
Trait 0 US *
TrAG1245.age.grp 2
TrAG1245 0 DIAG 0.0024 0.0019 0.0020 0.00026
age.grp 0 ID
TrSG123.sex.grp 2
TrSG123 0 DIAG 0.93 16.0 0.28
sex.grp 0 ID
xfa(Trait,1).sire 2
xfa(Trait,1) 0 XFA1 * !GP
sire 0 ID
xfa(TrDam123,1).dam 2
xfa(TrDam123,1) 0 XFA1 * !GP
dam 0 ID
xfa(TrLit1234,1).lit 2
xfa(TrLit1234,1) 0 XFA1 * !GP
lit 0 ID

#FUNCTIONAL
wwt ywt gfw fdm fat   Trait Trait.age Trait.brr Trait.sex Trait.age.sex !r $VARF,
xfa1(Trait).sire xfa1(TrDam123).dam xfa1(TrLit1234).lit ,
!f Trait.grp mv
residual id(units).us(Trait)

!PATH 3 #CHANGE XFA1 TO UNSTRUCTURED FOR SIRE AND LITTER
#STRUCTURAL
wwt ywt gfw fdm fat   Trait Trait.age Trait.brr Trait.sex Trait.age.sex !r $VARS,
Trait.sire xfa(TrDam123,1).dam TrLit1234.lit ,
!f Trait.grp mv
1 2 5 #1 R STRUCTURE WITH 2 COMPONENTS AND 5 G STRUCTURES
0 0 0 #INDEPENDENT ACROSS ANIMALS
Trait 0 US *
TrAG1245.age.grp 2
TrAG1245 0 DIAG 0.0024 0.0019 0.0020 0.00026
age.grp 0 ID
TrSG123.sex.grp 2
TrSG123 0 DIAG 0.93 16.0 0.28
sex.grp 0 ID
Trait.sire 2
Trait 0 US !GP *
sire 0 ID
xfa(TrDam123,1).dam 2
xfa(TrDam123,1) 0 XFA1 *
```

3.10 Multivariate animal genetics data - Sheep

```
dam 0 ID
TrLit1234.lit 2
TrLit1234 0 US *
lit 0 ID

#FUNCTIONAL
wwt ywt gfw fdm fat   Trait Trait.age Trait.brr Trait.sex Trait.age.sex !r $VARF,
us(Trait).sire xfa1(TrDam123).dam us(TrLit1234).lit ,
!f Trait.grp mv
residual id(units).us(Trait)



!PATH 3 #WORK OUT FUNCTIONS OF PARAMETERS FOR PATH 3
VPREDICT !DEFINE
X Damv 38:43 # defines 54:59
F phenWYG 1:6 + 23:28 + 54:59 + 44:49 # defines 60:65
F phenD 7:10 + 29:32 + 50:53 # defines 66:69
F phenF 11:15 + 33:37 # defines 70:74
F Direct 23:37 * 4.  # defines 75:89
F Maternal 54:59 - 23:28 # defines 90:95
F Resid 60:74 - 75:89 # defines 96:110
H WWTh2 75 60
H YWTh2 77 62
H GFWh2 80 65
H FDMh2 84 69
H FATh2 89 74
R GenCor 23:37
R MatCor 90:95


#STRUCTURAL
VPREDICT !DEFINE
X Damv xfa(TrDam123,1) # defines 54:59
F phenWYG Residual[1:6]+sire[1:6]+TrLit1234.lit[1:6]+xfa(TrDam123,1).dam
# defines 60:65= 1:6 + 23:28 + 44:49 + 54:59
F phenD Residual[7:10]+Trait.sire[7:10]+ TrLit1234.lit[7:10]
# defines 66:69= 7:10 + 29:32 + 50:53
F phenF Residual[11:15]+Trait.sire[11:15]
# defines 70:74= 11:15 + 33:37
F Direct Trait.sire *4.  #defines 75:  89= 23:37 * 4.
F Maternal Damv -Trait[1:6] #defines 90:  95= 54:59 - 23:28
F residWYG phenWYG - Trait.sire[1:6] #defines 96:101= 60:65 - 23:28
F residWYG phenD - Trait.sire[7:10] #defines 102:105= 66:69 - 29:32
F residWYG phenF - Trait.sire[11:15] #defines 106:110= 70:74 - 33:37
#defines 96:110= 60:74 - 23-37
H WWTh2 Direct[1] phenWYG[1] # 75 60
```

3.10 Multivariate animal genetics data - Sheep

```
H YWTh2 Direct[3] phenWYG[3] # 77 62
H GFWh2 Direct[6] phenWYG[6] # 80 65
H FDMh2 Direct[10] phenD[4] # 84 69
H FATh2 Direct[15] phenF[5] # 89 74
R GenCor Trait.sire # 23:37
R MatCor Maternal # 90:95


#FUNCTIONAL
VPREDICT !DEFINE
#USING !ASSIGN TO GIVE CONCISE VPREDICT
!ASSIGN lusT lit;us(TrLit1234) # us(TrLit1234).id(lit);us(TrLit1234)
!ASSIGN susT sire;us(Trait) #us(Trait).id(sire);us(Trait)
!ASSIGN uusT id(units).us(Trait);us(Trait)
X Damv xfa1(TrDam123) # defines 54:59
F phen $uusT[1:6]+$susT[1:6]+$lusT[1:6]+xfa1(TrDam123)
# defines [1:6] elements of phen
# defines 60:65= 1:6 + 23:28 + 44:49 + 54:59
F phen $uusT[7:10]+$susT[7:10]+ $lusT[7:10]
# defines [7:10] elements of phen
# defines 66:69= 7:10 + 29:32 + 50:53
F phen $uusT[11:15]+$susT[11:15]
# defines [11:15] elements of phen
# defines 70:74= 11:15 + 33:37
F Direct $susT *4.  #defines 75:  89= 23:37 * 4.
F Maternal Damv -$susT[1:6] #defines 90:  95= 54:59 - 23:28
F resid phen - $susT #defines 96:110= 60:74 - 23-37
H WWTh2 Direct[1] phen[1] #defines 111= 75/ 60
H YWTh2 Direct[3] phen[3] #defines 112= 77/ 62
H GFWh2 Direct[6] phen[6] #defines 113= 80/ 65
H FDMh2 Direct[10] phen[10] #defines 114= 84/ 69
H FATh2 Direct[15] phen[15] #defines 115= 89/ 74
R GenCor $susT #defines 116:125 from 23:37
R MatCor Maternal #defines 126:129 from 90:95


!RENAME 1 !ARG 3 #CHANGE 1 TO 2,3,4 OR 5 FOR OTHER PATHS
Multivariate Animal model
!DOPATH $1
tag !P
sire 92 !I
dam !P
grp 49
sex
brr 4
litter 4871
age
```

3.10 Multivariate animal genetics data - Sheep

```
wwt !MO # !MO identifies missing values
ywt !MO
gfw !MO
fdm !MO
fat !MO
pcoop.fmt # read pedigree from first three fields
!PATH 1 // pcoop.fmt
!PATH 2 // pcoop.fmt !CONTINUE !MAXI 40
!PATH 3 // pcoop.fmt !CONTINUE !MAXI 40
!PATH 4 // pcoop.fmt !CONTINUE !MAXI 40
!PATH 5 // pcoop.fmt !CONTINUE !MAXI 40



!PATH 0 #USING SUBSET TO ALLOW EASY TRAIT ASSOCIATION WITH FACTORS IN MODEL
!SUBSET TrDam12 Trait 1 2 0 0 0
!SUBSET TrLit1234 Trait 1 2 3 4 0
!SUBSET TrAG1245 Trait 1 2 4 5
!SUBSET TrSG123 Trait 1 2 3 0 0
!SUBSET TrDa123 Trait 1 2 3 0 0


#USING !ASSIGN TO MAKE SPECIFICATION CLEARER
#STRUCTURAL
!ASSIGN TDIAGI 2.3759 6.2256 0.60075E-01 0.63086 0.13069 !GP
!ASSIGN DDIAGI 2.1584 2.3048 !GP
!ASSIGN LDIAGI 3.55265 2.55777 0.191238E-01 0.897272 !GP
!ASSIGN RUSI !< !GP
13.390 9.0747 17.798 0.31961 0.87272 0.13452
0.71374 1.4028 0.23141 4.0677 0.72812 2.0831 0.75977E-01 0.25782 1.5337 !>
!ASSIGN VARS TrAG1245.age.grp TrSG123.sex.grp

#FUNCTIONAL
!ASSIGN TDIAGI !INIT 2.3759 6.2256 0.60075E-01 0.63086 0.13069 !GP
!ASSIGN DDIAGI !INIT 2.1584 2.3048 !GP
!ASSIGN LDIAGI !INIT 3.55265 2.55777 0.191238E-01 0.897272 !GP
!ASSIGN RUSI !< !INIT 13.390 9.0747 17.798 0.31961 0.87272 0.13452
0.71374 1.4028 0.23141 4.0677 0.72812 2.0831 0.75977E-01 0.25782 1.5337 !GP !>
!ASSIGN VARF !<
diag(TrAG1245 !INIT 0.0024 0.0019 0.0020 0.00026).age.grp,
diag(TrSG123 !INIT 0.93 16.0 0.28).sex.grp !>



!PATH 1 #USING DIAG FOR TAG,DAM AND LIT US FOR RESIDUAL
#STRUCTURAL
```

3.10 Multivariate animal genetics data - Sheep

```
wwt ywt gfw fdm fat   Trait Trait.age Trait.brr Trait.sex Trait.age.sex !r $VARS,
Trait.tag TrDam12.dam TrLit1234.lit ,
!f Trait.grp
1 2 5 #1 R STRUCTURE WITH 2 COMPONENTS AND 5 G STRUCTURES
0 0 0 #INDEPENDENT ACROSS ANIMALS
Trait 0 US $RUSI #UNSTRUCTURED TRAIT MATRIX INITIAL VALUES FROM UNIVARIATE ANALYSETrAG1245.age.grp
2
TrAG1245 0 DIAG 0.0024 0.0019 0.0020 0.00026
age.grp 0 ID
TrSG123.sex.grp 2
TrSG123 0 DIAG 0.93 16.0 0.28
sex.grp 0 ID
Trait.tag 2
Trait 0 DIAG $TDIAGI
tag 0 AINV
TrDam12.dam 2
TrDam12 0 DIAG $DDIAGI
dam 0 ID
TrLit1234.lit 2
TrLit1234 0 DIAG $LDIAGI
lit 0 ID

#FUNCTIONAL
wwt ywt gfw fdm fat   Trait Trait.age Trait.brr Trait.sex Trait.age.sex !r $VARF,
diag(Trait $TDIAGI).nrm(tag) diag(TrDam12 $DDIAGI).dam diag(TrLit1234 $LDIAGI).lit ,
!f Trait.grp
residual id(units).us(Trait $RUSI)

!PATH 2 #USING XFA1 FOR TAG,DAM AND LIT US FOR RESIDUAL
#STRUCTURAL
wwt ywt gfw fdm fat !SIGMAP   Trait Trait.age Trait.brr Trait.sex Trait.age.sex !r $VARS,
xfa(Trait,1).tag xfa(TrDam12,1).dam xfa(TrLit1234,1).lit ,
!f Trait.grp
1 2 5
0 0 0
Trait 0 US $RUSI
TrAG1245.age.grp 2
TrAG1245 0 DIAG 0.0024 0.0019 0.0020 0.00026
age.grp 0 ID
TrSG123.sex.grp 2
TrSG123 0 DIAG 0.93 16.0 0.28
sex.grp 0 ID
xfa(Trait,1).tag 2
xfa(Trait,1) 0 XFA1 * !GP
tag 0 AINV
xfa(TrDam12,1).dam 2 * !GP
xfa(TrDam12,1) 0 XFA1
```

```
dam 0 ID
xfa(TrLit1234,1).lit 2 * !GP
xfa(TrLit1234,1) 0 XFA1
lit 0 ID

#FUNCTIONAL
wwt ywt gfw fdm fat   Trait Trait.age Trait.brr Trait.sex Trait.age.sex !r $VARF,
xfa1(Trait).nrm(tag) xfa1(TrDam12).dam xfa1(TrLit1234).lit ,
!f Trait.grp
residual id(units).us(Trait)

!PATH 3
#STRUCTURAL
wwt ywt gfw fdm fat   Trait Trait.age Trait.brr Trait.sex Trait.age.sex !r $VARS,
Trait.tag xfa(TrDam12,1).dam TrLit1234.lit ,
!f Trait.grp
1 2 5
0 0 0
Trait 0 US !GP *
TrAG1245.age.grp 2
TrAG1245 0 DIAG 0.0024 0.0019 0.0020 0.00026
age.grp 0 ID
TrSG123.sex.grp 2
TrSG123 0 DIAG 0.93 16.0 0.28
sex.grp 0 ID
Trait.tag 2
Trait 0 US * !GP
tag 0 AINV
xfa(TrDam12,1).dam 2
xfa(TrDam12,1) 0 XFA1 * !GP
dam 0 ID
TrLit1234.lit 2
TrLit1234 0 US * !GP
lit 0 ID
#FUNCTIONAL
wwt ywt gfw fdm fat   Trait Trait.age Trait.brr Trait.sex Trait.age.sex !r $VARF,
us(Trait).nrm(tag) xfa1(TrDam12).dam us(TrLit1234).lit ,
!f Trait.grp
residual id(units).us(Trait)

!PATH 4 #FITTING XFA2 FOR TAG
#STRUCTURAL
wwt ywt gfw fdm fat   Trait Trait.age Trait.brr Trait.sex Trait.age.sex !r $VARS,
xfa(Trait,2).tag xfa(TrDam12,1).dam TrLit1234.lit ,
!f Trait.grp
1 2 5
0 0 0
```

3.10 Multivariate animal genetics data - Sheep

```
Trait 0 US !GP
TrAG1245.age.grp 2
TrAG1245 0 DIAG 0.0024 0.0019 0.0020 0.00026
age.grp 0 ID
TrSG123.sex.grp 2
TrSG123 0 DIAG 0.93 16.0 0.28
sex.grp 0 ID
xfa(Trait,2).tag 2
xfa(Trait,2) 0 XFA2 *!GP
tag 0 AINV
xfa(TrDam12,1).dam 2
xfa(TrDam12,1) 0 XFA1 * !GP
dam 0 ID
TrLit1234.lit 2
TrLit1234 0 US * !GP
lit 0 ID
#FUNCTIONAL
wwt ywt gfw fdm fat   Trait Trait.age Trait.brr Trait.sex Trait.age.sex !r $VARF,
xfa2(Trait).nrm(tag) xfa1(TrDam12).dam us(TrLit1234).lit ,
!f Trait.grp
residual id(units).us(Trait)

!PATH 5 #FITTING XFA3 FOR TAG
#STRUCTURAL
wwt ywt gfw fdm fat   Trait Trait.age Trait.brr Trait.sex Trait.age.sex !r $VARS,
xfa(Trait,3).tag xfa(TrDam12,1).dam TrLit1234.lit ,
!f Trait.grp
1 2 5
0 0 0
Trait 0 US !GP
TrAG1245.age.grp 2
TrAG1245 0 DIAG 0.0024 0.0019 0.0020 0.00026
age.grp 0 ID
TrSG123.sex.grp 2
TrSG123 0 DIAG 0.93 16.0 0.28
sex.grp 0 ID
xfa(Trait,3).tag 2
xfa(Trait,3) 0 XFA3 * !GP
tag 0 AINV
xfa(TrDam12,1).dam 2
xfa(TrDam12,1) 0 XFA1 * !GP
dam 0 ID
TrLit1234.lit 2
TrLit1234 0 US * !GP
lit 0 ID

!PATH 5
```

3.10 Multivariate animal genetics data - Sheep

```
#FUNCTIONAL
wwt ywt gfw fdm fat   Trait Trait.age Trait.brr Trait.sex Trait.age.sex !r $VARF,
xfa3(Trait).nrm(tag) xfa1(TrDam12).dam us(TrLit1234).lit ,
!f Trait.grp
residual id(units).us(Trait)
```

Bibliography

Wolfinger, R. D. (1996). Heterogeneous variance-covariance structures for repeated measures, *Journal of Agricultural, Biological, and Environmental Statistics* **1**: 362–389.