
ASReml Update

What's new in Release 3
2009

A R Gilmour
B R Cullis
R Thompson

NSW Department of Industry and Investment, Orange, Australia

School of Mathematical Sciences, Queen Mary, University of London, Mile End
Road, London E1 4NS, and Centre for Mathematical and Computational Biology,
and Department of Biomathematics and Bioinformatics, Rothamsted Research,
Harpenden AL5 2JQ, United Kingdom

ASReml Update. What's new in Release 3.00

A R Gilmour, 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.vsnl.co.uk/>

Copyright Notice

Copyright © 2009, NSW Department of Industry and Investment. All rights reserved.

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

The correct bibliographical reference for this document is:

Gilmour, A.R., Cullis, B.R. and Thompson, R. 2009 ASReml Update: What's new in Release 3.00 VSN International Ltd, Hemel Hempstead, HP1 1ES, UK

Author email addresses

Arthur.Gilmour@Cargovale.com.au
Brian.Cullis@industry.nsw.gov.au
Robin.Thompson@bbsrc.ac.uk

Preface

ASReml, a statistical package that fits linear mixed models using Residual Maximum Likelihood (REML), is a joint project of NSW Department of Primary Industries¹ and Rothamsted Research². It provides a stable platform for delivering well established procedures while also delivering current research in the application of linear mixed models. The strength of ASReml is the use of the Average Information (AI) algorithm and sparse matrix methods for fitting the linear mixed model. This enables it to analyse large and complex data sets quite efficiently.

This document highlights the developments in ASReml since Release 2.00 and is intended as a transition document for existing users. New users should refer to ASReml User Guide, Release 3.00.

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

The stand-alone ASReml program is one of several user interfaces to the underlying computational engine. Genstat uses the same engine in its REML directive and the `asrem1` class of functions available for R (`asrem1-r`) also use the same engine. Both of these have good data manipulation and graphical facilities. `asrem1-r` users may use the stand-alone version with the same license file, and vice versa.

The focus in developing stand-alone ASReml has been on the core engine and it is freely acknowledged that its user interface is not to the level of these other packages. Nevertheless, as the developers interface, it is functional, it gives access

¹Kite St, Orange, New South Wales, 2800, Australia

²Harpenden, Hertsfordshire, United Kingdom

to everything that the core can do and is especially suited to batch processing and running of large models without the overheads of other systems. Feedback from users is welcome and attempts will be made to rectify identified problems in ASReml.

Briefly, the improvements in Release 3.00 include

- Ability to specify the exact form of the conditional Wald F statistic formed (!FOWN),
- Enhanced prediction facilities for heirarchal factor structures (!ASSOCIATE),
- More facility in handling Marker data and using IBD matrices,
- New pedigrees options for sex linked traits and inbred lines (!FGEN, !XLINK),
- Outlier detection diagnostics - outlier statistics for random terms (including the residual) (!OUTLIER),
- Multinomial distribution: analysis of ordinal data under a threshold model (!MULTINOMIAL),
- Syntax for defining nested R structures
- Facility for merging data
- Improved facilities for cycling through a large number of covariates and models,
- Forming BLUPs using Preconditioned Conjugate Gradient and Gauss-Seidel iteration

The data sets and ASReml input files used in this guide are available from <http://www.vsnl.co.uk/software/asreml> as well as in the `examples` directory of the distribution CD-ROM. They remain the property of the authors or of the original source but may be freely distributed provided the source is acknowledged.

Proceeds from the licensing of ASReml are used to support continued development to implement new developments in the application of linear mixed models. The developmental version is available to supported licensees via a website upon request to VSN. Most users will not need to access the developmental version unless they are actively involved in testing a new development.

Acknowledgements

We gratefully acknowledge the Grains Research and Development Corporation of Australia for their financial support for our research since 1988. Brian Cullis and Arthur Gilmour thank the NSW Department of Primary Industries for providing a stimulating and exciting environment for applied biometrical research and consulting. Rothamsted Research receives grant-aided support from the Biotechnology and Biological Sciences Research Council of the United Kingdom.

We sincerely thank Ari Verbyla, Sue Welham, Dave Butler and Alison Smith, the other members of the ASReml ‘team’. Ari contributed the cubic smoothing splines technology, information for the Marker map imputation, on-going testing of the software and numerous helpful discussions and insight. Sue Welham has overseen the incorporation of the core into Genstat and contributed to the `predict` functionality. Dave Butler has developed the `asreml-r` class of functions. Alison contributed to the development of many of the approaches for the analysis of multi-section trials. We also thank Ian White for his contribution to the spline methodology. The Matérn function material was developed with Kathy Haskard, a PhD student with Brian Cullis, and the denominator degrees of freedom material was developed with Sharon Nielsen, a Masters student with Brian Cullis. Damian Collins contributed the PREDICT !PLOT material and has reviewed the Generalised Linear (Mixed) Models implementation. Greg Dutkowski has contributed to the extended pedigree options. Chandrapal Kailasanathan contributed to the new MERGE facility. The `asremlload.dll` functionality is provided under license to VSN. Alison Kelly has helped with the review of the XFA models. Finally, we especially thank our close associates who continually test the enhancements.

Arthur Gilmour acknowledges the grace of God through Jesus Christ our Saviour (*In Him are hidden all the treasures of wisdom and knowledge* Colossians 2:3)



Figure 1 Brian Cullis, Arthur Gilmour, David Butler

Contents

Preface	ii
1 Introduction	1
2 New features in ASReml 3	3
2.1 !Assign	3
2.2 Factor definition	4
!AS declaration	4
!G declaration	4
2.3 Transformations	4
Drop transformation	4
DO/ENDDO transformation	5
NA transformation	6
2.4 Pedigree options	6
Sex linked relationship matrix	6
Pedigree qualifiers	6
Performance in terms of speed for large pedigrees	9

Genetic groups in GIV matrices	9
2.5 Data File line qualifiers	10
Working Directory	10
!FOWN	10
!GROUPFACTOR	12
!SUBGROUP	12
More MyBasisFunction options	12
!HOLD	13
Forming BLUPs	14
2.6 Model Terms	15
vect covariate	15
Selecting a single variable from a group	15
2.7 !CYCLE command	15
2.8 Double slash	17
2.9 Multivariate data presentation	18
2.10 Multinomial Ordinal Multiple Threshold Models	18
2.11 Nested R structure	21
Equating variance structures	21
2.12 PREDICT extension	22
Associated factors	23
2.13 VPREDICT: PIN file processing	26

2.14 Iterative Schemes	26
2.15 !CONTINUE	27
XFA extensions	27
2.16 Residuals	28
.vll file	28
Alternative Outlier Model	28
3 Command file: Merging data files	30
3.1 Introduction	31
3.2 Merge Syntax	31
3.3 Examples	33
Bibliography	33
Index	34

ASReml was developed in 1996 following joint research by its principle authors. It was distributed free from the Rothamsted website for five years as it matured into a significant provider of a broad range of linear mixed models.

ASReml 1 was released commercially in 2002 through VSN International, the company established by Rothamsted and NAG to maintain and distribute Genstat. The relationship of Genstat and ASReml has always been close with the core REML routines from ASReml becoming the AI REML engine in Genstat in about 1999.

ASReml 2 was released early in 2007 with substantial refinements. Possibly the most significant from the user perspective being the calculation of Denominator degrees of freedom so that Wald F statistics can be formally tested. This is the basis for the new Discovery ASReml (<http://www.vsnl.co.uk/software/asrem1/>).

ASReml 3 is now released. The core functionality and output structures have not been changed, but numerous small changes have been made to facilitate the use of mixed models in a wide range of applications. This document outlines the principal changes which include

- Ability to specify the exact form of the conditional Wald F statistic formed (!FOWN),
- Enhanced prediction facilities for heirarchal factor structures (!ASSOCIATE),
- More facility in handling Marker data and using IBD matrices,
- New pedigrees options for sex linked traits and inbred lines (!FGEN, !XLINK),
- Outlier detection diagnostics - outlier statistics for random terms (including the residual) (!OUTLIER),
- Multinomial distribution: analysis of ordinal data under a threshold model (!MULTINOMIAL),

- Syntax for defining nested R structures
- Facility for merging data
- Improved facilities for cycling through a large number of covariates and models,
- Forming BLUPs using Preconditioned Conjugate Gradient and Gauss-Seidel iteration

2

New features in ASReml 3

2.1 !Assign

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, on a line by itself.

The syntax is, beginning in position 1,

```
!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.

For example

```
!ASSIGN TRT xfa(Treat,1)
...
... $TRT.geno
...
$TRT.geno 2
$TRT 0 XFA1
...
geno
```

Restrictions

- A maximum of 20 assign strings may be defined.
- The combined length of all strings is 1000 characters.
- *name* may consist of 1–4 characters but should not begin with a number (see command line arguments).

- Dollar substitution occurs before most other high level actions. Consequently, ASSIGN strings and commandline arguments may substitute into a !CYCLE line.

2.2 Factor definition

!AS declaration

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

Diallel experiments are common in some species. The usual way of fitting these in ASReml is to declare the two parents in two factors say `Male` and `Female`, and to fit the general combining effect in the model as `Male and(Female)` so that the design matrix for `Female` is overlaid or (added to) the design matrix for `Male`. This requires that the two factors be coded conformable so that the i th 'male' (parent) is the same individual as the i th 'female' (parent). If they are not directly coded that way, a pedigree file can be used, but the new !AS qualifier is more convenient. Thus

```
Male !A
Female !AS Male
```

codes `Male` and `Female` using to a common integrated list of factor level labels for the two factors.

!G declaration

The !G m declaration which declares a set of m variables to be treated as a set of m variates has been extended so that the !G $m n$ declaration declares a set of m variables be treated as a set of m factors with n levels each.

2.3 Transformations

Drop transformation

- The !D[o] v transformation, conditional on logical operator o , discards records depending on the value of its argument and the value in the *test*

field. It drops the record if the test generates TRUE or the test field is missing.

- The new `!DV[o] v` transformation discards records, subject to the logical operator *o*, which have *v* in the field but keeps records with 'missing value' in the field unless the test value is `*`. That is `!D >99` is equivalent to `!DV * !DV >99`. Use
 - `!DV *` to discard records with a missing value in the test field,
 - `!DV v` to discard records with a *v* in the test field,
 - `!DV<v` to discard records when the test field has a value $< v$,
 - `!DV<=v` to discard records when the test field has a value $\leq v$,
 - `!DV<>v` to discard records when the test field has a value $\neq v$,
 - `!DV>=v` to discard records when the test field has a value $\geq v$,
 - `!DV>v` to discard records when the test field has a value $> v$.

DO/ENDDO transformation

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

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

is equivalent to

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

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

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

NA transformation

The NA v transformation has been extended to allow v to refer to another field instead of just being a value. This facilitates copying a value from another field when the current field has a missing value. e.g. `WT1 WT2 WT !=WT2 !NA WT1` defines WT with values of WT2, or WT1 if WT2 is missing.

2.4 Pedigree options

Sex linked relationship matrix

Fernando and Grossman (1990) described formation of a relationship matrix for the X chromosome in species where the male is XY and the female is XX. This (inverse) has been added as an option (`!XLINK`) when the usual additive (inverse) relationship matrix is formed.

A subroutine based on the method of Sargolzaei *et al.* (2005) has been implemented as `!METHOD 4`. Method 4 is ideal for large deep pedigrees.

Pedigree qualifiers

ASReml pedigree qualifiers fall into three classes. They are listed here with a description where the qualifier is new or has been extended or modified.

- Those related to parsing the pedigree file

!SKIP n	heading lines to skip
!ALPHA [c]	Alphanumeric identifiers up to c (default 20) characters. The ability to specify the number of characters is new proposal (not yet implemented).
!LONGINTEGER	Long (16 digit) integer are indicated by this qualifier. The default (without !LONGINTEGER or !ALPHA) is integer identifiers of up to 8 digits.
!MAKE	reforms the \mathbf{A}^{-1} matrix each run. Otherwise ASReml will retrieve the inverse from the <code>ainverse.bin</code> file if it is present.
!QUAAS	The original routine for calculating \mathbf{A}^{-1} in <code>.asrwas</code> based on Quaas (1976)
!MEUWISSEN	The default method for forming \mathbf{A}^{-1} is based on the algorithm of Meuwissen and Luo (1992).
!SARGOLZAEI	A third method for computing \mathbf{A}^{-1} was developed by Sargolzaei <i>et al.</i> (2005).
!REPEAT	allows repeat identifiers
!SORT	sorts the pedigree into birth order and writes it out to a new file.

- Those relating to output
 - !DIAG writes names, inbreeding, $\text{diag}(\mathbf{A}^{-1})$ to the file `base-name.aif` (formerly written to `AINVERSE.DIA`).
 - !GIV writes \mathbf{A}^{-1} to `basename.A.giv` (formerly written to `AINVERSE.GIV`) in GIV format.
- Those related to forming the genetic matrices.
 - !FGEN [f] indicates the pedigree file contains a fourth field indicating the generation of Selfing or the level of inbreeding in a base individual. In the fourth field, 0 indicates a simple cross, 1 indicates selfed once, 2 indicates selfed twice, etc.. A value between 0 and 1 for a base individual is taken as its inbreeding value. If the pedigree has implicit individuals (they appear as parents but not in the first field of the pedigree file), they will be assumed base non-inbred individuals unless their inbreeding level is set with !FGEN f where $0 < f < 1$ is the inbreeding level of such individuals.

- !GOFFSET** *o* An alternative to group constraints described above is to shrink the group effects by adding the constant *o* to the diagonal elements of \mathbf{A}^{-1} pertaining to groups. When a constant is added, no adjustment of the degrees of freedom is made for genetic groups. Use **!GOFFSET -1** to suppress adding of constraints where empty groups appear. The empty groups are then not counted in the DF adjustment.
- !GROUPS** *g* The first *g* pedigree lines define genetic groups. You may insert Groups with no members to define constraints on groups. A constraint is added to the inverse which causes the preceding set of groups which have members to have effects which sum to zero. The issue is to get the degrees of freedom correct and to get the correct calculation of the Likelihood, especially in bivariate cases where DF associated with groups may differ between traits. The **!LAST** qualifier is designed to help as without it, reordering may associate singularities in the \mathbf{A} matrix with random effects which at the very least is confusing. When the \mathbf{A} matrix incorporates fixed effects, the number of DF involved may not be obvious, especially if there is also a sparsely fitted fixed HYS factor. The number of Fixed effects (degrees of freedom) associated with GROUPS is taken as the declared number less twice the number of constraints applied. This assumes all groups are represented in the data, and that degrees of freedom associated with group constraints will be fitted elsewhere in the model.
- !INBRED** Assumes all pedigree individuals are fully in bred. This option may not be used with **!FGEN**.
- !MGS** Maternal Grandsire model
- !SELF** *s* assumes a proportion *s* of individuals with unknown 'dam' are selfed.
- !XLINK** requests the formation of the (inverse) relationship matrix for the X chromosome. This will typically be accessible as GIV1 or as specified in the output.

The extension options are available with the MEUWISSEN method for \mathbf{A}^{-1} and generally may not be used in combination.

Performance in terms of speed for large pedigrees

The methods of computing inverse of A were tested on three pedigrees. The pedigrees had 32000 members and are described as types:

1. All related to individual 1 by randomly selecting parents from the whole extant population.
2. First fifty individuals were base animals, matings at random across the population.
3. First 50 individuals were base animals. Next fifty individuals were descendants of previous fifty individuals (generation by generation information).

	Method 4 !SARGOLZAEI	Method 3 !MEUWISSEN	Method 2 !QUASS
No genetic line/ generation info.	9.496s	5.586s	11.261s
Genetic lines	5.942s	6.051s	11.203s
Generation by generation	19.82s	72.22s	20.79s

Method 4 performed faster with sparse matrices created when genetic line/ generation by generation information are included. Method 3 performed quicker with dense matrices created when no genetic line/ generation by generation information are included.

Genetic groups in GIV matrices

A `!GROUPSDF n` qualifier has been added to the GIV matrix specification line to allow for adjustment of any fixed degrees of freedom incorporated into the GIV matrix. This enables a GIV matrix generated using a pedigree with the `!GROUPS` qualifier to be used again as a GIV matrix. The value of the argument is the number of degrees of freedom fitted by the GIV matrix which will normally be the number of groups.

When groups are constrained, then it will be the number of groups less number of constraints. For example, if the pedigree file qualified by `!GROUPSDF 7` begins

```
A 0 0
```

```

B 0 0
C 0 0
ABC 0 0 # ABC is not present in the subsequent pedigree lines
D 0 0
E 0 0
DE 0 0 # DE is not present in the subsequent pedigree lines

```

there are actually only 5 genetic groups and two constraints so that the fixed effects for A, B and C sum to zero, and for D and E sum to zero so actually only 3 fixed degrees of freedom are fitted. Therefore if the **A** inverse for this pedigree was saved, and subsequently used as a GIV matrix, it should be declared as `!GROUPDF 3`.

ASReml 3 writes this qualifier to the top of the `A.giv` file when appropriate, and checks there when reading a `.giv` file so that the user does not need to explicitly include this qualifier in the `.as` file when the `A.giv` file was produced with ASReml 3.

2.5 Data File line qualifiers

Working Directory

The path to the folder containing the data may be incorporated into the data file name or specified in a separate `!FOLDER` qualifier inserted BEFORE the data file line. Thus `"\Data Folder\data.asd"`

is equivalent to

```

!FOLDER "\Data Folder"
data.asd

```

!FOWN

The `!FOWN` qualifier may be used to control F-tests reported under the `F-con` heading. It has the form

```
!FOWN terms to test ; background terms
```

placed on a separate line immediately before or after the model line. Multiple `!FOWN` statements should appear together. It generates an F-test statistic for each model term in *terms to test* which tests its contribution after all after terms in *terms to test* and *background terms*, conditional on all terms that appear in the SPARSE equations, and on not changing the degrees of freedom associated with

a term. It only needs to include terms which will appear in the ANOVA table.

For example,

```
!FOWN A B C ; mu
!FOWN A.B B.C A.C ; mu A B C
!FOWN A.B.C ; mu A B C A.B B.C A.C
```

would request the tests

```
R(A | mu B C sparse),
R(B | mu A C sparse),
R(C | mu A B sparse),
R(A.B | mu A B C B.C A.C sparse),
R(B.C | mu A B C A.B A.C sparse),
R(A.C | mu A B C A.B B.C sparse) and
R(A.B.C | mu A B C A.B A.C B.C sparse).
```

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

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

and

$$y \sim 1 + A$$

Warnings: This qualifier is provided for advanced users who have a good understanding of marginality issues in ASReml. ASReml does not verify the tests requested satisfy marginality considerations which are normally relevant. Any model terms in the !FOWN lists which do not appear in the actual model, are ignored without flagging an error. Any model terms which are omitted from !FOWN statements are tested with the usual conditional test. If any model terms are listed twice, only the first test is performed. F-con tests specified in !FOWN statements are given model codes 0, P,

The !FOWN statements are parsed by the same routine that parses the model line and so accepts the same model syntax options. Care should be taken to ensure term names are consistently spelt. If the !FOWN statements appear before the model line, model terms that are not previously defined may not be abbreviated (truncated) in !FOWN statements relative to their form in the model line because they are defined on their first appearance.

!GROUPFACTOR

The !GROUPFACTOR qualifier, like !SUBSET, must appear on a line by itself after the data line and before the model line. Its purpose is to define a factor by merging levels of an existing factor. The syntax is

```
!GROUPFACTOR <Group_factor> <Exist_factor> <new codes>
```

for example

```
!GROUPFACTOR Year YearLoc 1 1 1 2 2 3 3 3 4 4
```

forms a new factor `Year` with 4 levels from the existing factor `YearLoc` with 10 levels.

!SUBGROUP

```
!SUBGROUP t v p
```

forms a new group factor (t) derived from an existing group factor (v) by selecting a subset (p) of its variables. It is analgous to the !SUBSET qualifier except that it applies to !G groups of variables.

More MyBasisFunction options

```
!SPARSE
```

`mbf(Term,lev)` has been extended to allow specification of a large SPARSE set of covariates ('basis functions'). The extension is activated by specifying !SPARSE on the !MBF line. i.e.

```
!MBF mbf(gen,450) sparsegen.mbf !sparse
```

The design to be used is then specified for each unique value of `gen`, in sorted (increasing) order, each row starting on a new line and consisting of `key` and then as many `column,value` pairs as required to specify the non zero elements of the design for that value of `key`. The pairs should be arranged in increasing order of `column` within rows. The rows may be continued on subsequent lines of the file provided incomplete lines end with a COMMA.

```
!RENAME newname
```

allows the `mbf()` term to be renamed. This is particularly useful when defining several `mbf` factors which would all have the same default name. For example

```
!MBF mbf(entry,3) mlib\m35.csv !rename Marker35
```

Automatic setting of `Lev` in `mbf(Term)`

`mbf(,)` has also been enhanced to allow

```
!MBF mbf(entry) entry.mbf
```

where the number of columns/covariates is determined from the file. This enhancement is not active with the `!SPARSE` option.

```
!KEY k !NOKEY !FIELD v !RFIELD v
```

allows the mbf factor to be defined from specified columns of the mbf covariate file. For example, the file (say `markers.csv`) may have covariate values for 400 markers in columns 2:401 with the variety key in field 1. A specific marker covariate can then be extracted as

```
!MBF mbf(variety,1) markers.csv !key 1 !FIELD 36 !rename Marker35
```

This is a bit awkward if the field number (36) and marker number (35) are to come from a substitution variable since they are different numbers. If the key field would just contains the numbers 1: n in order, because there are n lines in the file corresponding to the n levels of the reference variable (`variety`) it may be omitted and the `!NOKEY` qualifier specified. `!NOKEY` implies that the key is implicit (1: n corresponding to the n data lines in the file). The default if neither `!KEY` or `!NOKEY` is specified is that the key field is present as the first field. An alternative to `!FIELD` is `!RFIELD` standing for *Relative Field*. The data field is then relative to the key field so that `Marker35` can be obtained as

```
!MBF mbf(variety,1) markers.csv !key 1 !RFIELD 35 !rename Marker35
```

Restrictions:

The key field MUST be numeric. In particular, if the data field it relates to is either an `!A` or `!I` encoded factor, the original (uncoded) level labels may not specified in the MBF file. Rather the coded levels must be specified. The MBF file is processed before the data file is read in and so the mapping to coded levels has not been defined in ASReml when the MBF file is processed, although the user can/must anticipate what it will be.

Comment:

If this MBF process is to be used repeatedly, it will generally be much faster processing in ASReml if the markers were written to separate files. ASReml will read 10 files containing a single field much faster than reading a single file containing 400 fields, ten times to extract 10 different markers.

!HOLD

As another mechanism to try when fitting complex variance models, use `!HOLD <list>` to temporarily fix the parameters listed. Parameter numbers have been added to the reporting of input values to facilitate use of this and other parameter

number dependent qualifiers. The list should be in increasing order using colon to indicate a sequence with step size 1. E.g. !HOLD 1:20 30:40.

Forming BLUPs

`!BLUP n`

is used to calculate the effects reported in the `.sln` file without calculating any derived quantities such as predicted values or updated variance parameters. For argument values 1:3, `.asr` solves for the effects directly while for values 4:19 it solves the mixed model equations by iteration, allowing larger models to be fitted. With direct solution, the estimation REML iteration routine is aborted after

$n = 1$: forming the estimates of the vector of fixed and random effects by matrix inversion,

$n = 2$: forming the estimates of the vector of fixed and random effects, REML log-likelihood and residuals (this is the default),

$n = 3$: forming the estimates of the vector of fixed and random effects, REML log-likelihood, residuals and inverse coefficient matrix.

For arguments 4, 10:19, `.asr` forms the mixed model equations and solves them iteratively to obtain solutions for the fixed and random effects. The options are:

$n = 4$: forming the estimates of the vector of fixed and random effects using the Preconditioned Conjugate Gradient (PCG) Method (Mrode, 2005),

$n = 10:19$ forming the estimates of the vector of fixed and random effects by Gauss-Seidel iteration of the mixed model equations, with relaxation factor $n/10$, The default maximum number of iterations is 12000. This can be reset by supplying a value greater than 100 with the `!MAXIT` qualifier in conjunction with the `!BLUP` qualifier. Iteration stops when the average squared update divided by the average squared effect is less than $1e^{-10}$.

Gauss-Seidel iteration is generally much slower than the PCG method.

`.asr` prints its standard reports as if it had completed the iteration normally, but since it has not completed it, some of the information printed will be incorrect. In particular, variance information on the variance parameters will always be unavailable. Standard errors on the estimates will be wrong unless $n=3$. Residuals are not available if $n=1$. Use of $n=3$ or $n=2$ will halve the processing time when compared to the alternative of using `!MAXIT 1` rather than a `!BLUP n` qualifier. However, `!MAXIT 1` does result in complete and correct output.

2.6 Model Terms

vect covariate

`vect(TVC)` (vec transpose) is used in a multivariate analysis on a multivariate set of covariates (v) to pair them with the variates. The test example included

```
signal !G 93 # 93 slides
background !G 93
dart.asd !ASUV
signal ~ Trait Trait.vect(background) ...
```

to fit a slide specific regression of `signal` on `background`. In this example, `signal` is a multivariate set of 93 variates and `background` is a set of 93 covariates. The signal values relate to either the Red or Green channels. So for each slide and channel, we need to fit a simple regression of `signal ~ mu background`. But the data for the 93 slides is presented in parallel. If it were presented in series, with a factor `slide` indexing the slides, the equivalent model would be

```
signal ~ slide slide.background.
```

Selecting a single variable from a group

Curly or square brackets may be used to fit a single variable out of a group of variables as in the following example which fits the 33 Marker covariates 1 at a time.

```
Marker screen
animal !P
Phenotype
Markers !G 33
!CYCLE 1:33
data.ped
data.asd
Phenotype ~ mu !r animal Markers{I}
```

2.7 !CYCLE command

Formerly, all `CYCLE` arguments needed to be listed on a single line (up to 4000 characters). Now, the list may be spread over several lines provided each incomplete list has a trailing `COMMA`.

Previously, a !JOIN quaifier was required to put the outputs from the various cycles into one file. Now !JOIN is implicit in !CYCLE.

Previously !CYCLE would not work in conjunction with the command line combination !RENAME !ARG. Now !CYCLE works as an inner loop to !RENAME !ARG. So now for example, you can set up a series of PATHS, use !RENAME !ARG to loop through them and use !CYCLE to loop through a set of dependent variables.

An item in the form $i:j$ where i and j are integers, is expanded with step size of 1. For example

```
!CYCLE 1:10 12:14 21 ,
      25:27
```

is equivalent to !CYCLE 1 2 3 4 5 6 7 8 9 10 12 13 14 21 25 26 27

A cycle string may consist of up to 4 substrings, separated by a semicolon and referenced as \$I \$J \$K and \$L respectively. For example

```
!CYCLE Y1;X1 Y2;X2
$I ~ mu $J
```

When cycling is active, an extra line is written to the .asr file containing some details of the cycle in a form which can be extracted to form an analysis summary by searching for LogL:. A heading for this extra line is written in the first cycle. For example

```
LogL:   LogL  Residual  NEDF   NIT  Cycle  Text
LogL: -208.97  0.703148   587     6  1466  "LogL Converged"
```

The LogL: line with the highest LogL value is repeated at the end of the .asr file.

Extended Example

The following code will run through 1000 models fitting 1000 different marker variables to some data. For processing efficiently the 1000 marker variables are held in 1000 separate files in subfolder MLIB and indexed by Genotype.

Marker screen

```
Genotype *
```

```

yield
PhenData.txt
!CYCLE 1:1000
!MBF mbf(Genotype) MLIB\Marker$I.csv !rename Marker$I
yld ~ mu !r Marker$I

```

Having completed the run, the Unix command sequence

```
grep LogL: screen.asr | sort > screen.srt
```

sorts a summary of the results to identify the best fit. The best fit can then be added to the model and the process repeated. Assuming `Marker35` was best, the revised job could be

```

Marker screen
Genotype *
yield
PhenData.txt
!CYCLE 1:1000
!MBF mbf(Genotype) MLIB\Marker$I.csv !rename Marker$I
!MBF mbf(Genotype) MLIB\Marker35.csv !rename MKR035
yld ~ mu !r MKR035 Marker$I

```

We have given `Marker35` a new name because the old name is still being generated by the `!CYCLE` unless it is modified to read

```
!CYCLE 1:34 36:1000 .
```

2.8 Double slash

A double slash (`//`) in the `.as` file causes following information to be treated as if it was on a new line. The `#` (comment) operator takes precedence.

For example, the code

```

row row AR1 0.1 !S2=2.1
col col AR1 0.1
row row AR1 0.1 !S2=2.2
col col AR1 0.1
row row AR1 0.1 !S2=2.2
col col AR1 0.1
site.geno 2
site 0 US !GP

```

```
6*0
geno
```

can now be written as

```
row row AR1 0.1 !S2=2.1 // col col AR1 0.1
row row AR1 0.1 !S2=2.2 // col col AR1 0.1
row row AR1 0.1 !S2=2.2 // col col AR1 0.1
site.geno 2 //site 0 US !GP //6*0 // geno
```

2.9 Multivariate data presentation

ASReml will only allow up to 20 dependent terms to be nominated but these may be grouped !*Gn* sets of variates so that more than 20 variates may be analysed (usually in conjunction with !ASUV). (See the `vect` example on page 15).

2.10 Multinomial Ordinal Multiple Threshold Models

ASReml 3.0 can analyse multiple threshold data for grouped or ungrouped categorical data. For ungrouped data, the dependent variable will typically be a single variable containing class scores. For example, the analysis of lodging score (1:4) in a variety trial.

```
This is the analysis for az06
dum
rep 4
column 2
row 36
variety 18 !A %!D1 !D3 !D9 !D11 !D13 !D15 !D18 !D10
weed !-0.3472
lodging
p95days
yield
az06.asd !SKIP 1
lodging !MULTINOMIAL 4 !ORDINAL ~ Trait variety !r rep
PREDICT variety
```

The threshold model is internally fitted as a cumulative multivariate model. The

'traits' analysed should be counts in the ordered categories. These are summed internally to form the cumulative distribution which is modelled as logit (!LOGIT), probit (!PROBIT) or Complementary LogLog (!CLOG) variables. The !MULTINOMIAL k !ORDINAL qualifiers request a threshold analysis be fitted for the k classes in the response variable. The model fits $t = k - 1$ ordered thresholds in the Trait model term.

With ungrouped data (above), the response variable will be a 'factor' type variable with levels in the ordinal order and observations in each class.

An alternative coding is analyse just t classes, specify t with the !THR qualifier and use the !TOTAL n qualifier to imply the k th class (analogous to grouped binomial data).

Predicted values are reported for the cumulative proportions.

The following example is the analysis of grouped data. Lambs were scored into 5 ordered classes on the basis of the conformation of their feet. Classes 1 and 2 rarely occurred and have been combined with class 3 for this analysis. The final classes are here called GoodFS, PoorFS and BadFS for classes 5, 4, 3 respectively. The ASReml coding to fit a two threshold model to the shape data is

ALWAN Lamb data from Gilmour 1983 thesis page 177-8

```
Year !L 1980 1981
Group !L Perendale_80 Boor_Romney_80 Booroola_80,
      Perendale_81 Boor_Romney_81
SEX SIRE !I 34
Total # Lambs in Sex.Group class
GoodFS PoorFS Scald Rot
BadFS !=Total !=GoodFS !=PoorFS
lamb.dat !skip 1 !DDF -1

GoodFS PoorFS BadFS !multinomial 3 !ordinal ~ Trait,
SEX Group !r SIRE .16783
```

The summary of this analysis is:

6 LogL=-105.627	S2= 1.0000	129 df	Dev/DF= 0.9683
7 LogL=-105.627	S2= 1.0000	129 df	Dev/DF= 0.9683
Deviance from GLM fit		129	124.91
Variance heterogeneity factor [Deviance/DF]			0.97

-- Results from analysis of GoodFS PoorFS --
 Notice: While convergence of the LogL value indicates that the model
 has stabilized, its value CANNOT be used to formally test differences
 between Generalized Linear (Mixed) Models.

Source	Model	terms	Gamma	Component	Comp/SE	% C
SIRE	34	34	0.174698	0.174698	2.80	0 P

Analysis of Variance	NumDF	DenDF	F_inc	Prob
11 Trait	2	77.8	405.40	<.001
3 SEX	1	129.0	5.61	0.019
2 Group	4	30.0	8.03	<.001

Notice: The DenDF values are calculated ignoring fixed/boundary/singular
 variance parameters using numerical derivatives.

Warning: This Analysis of Variance based on the working variable is not
 equivalent to the Analysis of Deviance. Standard errors are scaled
 by the variance of the working variable, not the residual deviance.

	Estimate	Standard Error	T-value	T-prev
2 Group				
2	-0.727154	0.273337	-2.66	
3	-1.76491	0.356574	-4.95	-2.93
4	-1.19399	0.273169	-4.37	1.61
5	-0.915605	0.242677	-3.77	1.16
3 SEX				
1	-0.197719	0.856093E-01	-2.31	
11 Trait				
1	1.54993	0.200126	7.74	
2	3.82051	0.216315	17.66	27.12
4 SIRE				
			34 effects fitted	

----- 1 -----
 SEX evaluated at 0.5000
 SIRE terms are ignored unless specifically included
 The cells of the hypertext are calculated from all model terms constructed
 solely from factors in the averaging and classify sets.

Group	Trait	Logit_value	Stand_Error	Ecode	Retransformed_value	approx_SE
1.	GoodFS	1.4511	0.1952	E	0.8102	0.0318
1.	PoorFS	3.7217	0.2114	E	0.9764	0.0054
2.	GoodFS	0.7239	0.1915	E	0.6735	0.0434
2.	PoorFS	2.9945	0.2064	E	0.9523	0.0103
3.	GoodFS	-0.3138	0.2986	E	0.4222	0.0707
3.	PoorFS	1.9567	0.3048	E	0.8762	0.0370
4.	GoodFS	0.2571	0.1913	E	0.5639	0.0475
4.	PoorFS	2.5277	0.2043	E	0.9261	0.0153
5.	GoodFS	0.5355	0.1444	E	0.6308	0.0342
5.	PoorFS	2.8060	0.1631	E	0.9430	0.0094

SED: Overall Standard Error of Difference 0.2871

Notice that while the data is of exclusive categories, the model is fitted on cumulative categories so for example, in the predicted values, 0.8102 is the proportion in GoodFS for Group 1, 0.9764 is the proportion in GoodFS+PoorFS.

2.11 Nested R structure

!SUBSECTION v on a variance structure line in the R structures block of the coding allows many independent blocks of correlated observations to be modelled with common variance and correlation parameters. The observations need to be sorted on a variable which defines the blocks. The blocks can be of different sizes. Any homogeneous variance correlation model defined in Table 7.3 of the ASReml3 User Guide may be used for the variance structure. This extends the R structure definition $\mathbf{R} = \oplus_{i=1}^s \mathbf{R}_i$ where $\mathbf{R}_i = \otimes_{j=1}^c \Sigma(\phi_{ij})$ such that $\Sigma(\phi_{i1})$ may have direct sum structure with common parameters. So, for generic times

```
1 1 0 # data sorted bids within auctions
```

```
0 0 AR1 0.5 !SUBSECTION auction
```

and for explicit times

```
1 1 0 # data sorted date within plot
```

```
0 date EXP 0.2 !SUBSECTION plot
```

Notice the leading zero standing for the total number of records across all auctions.

Equating variance structures

In some plant breeding applications, it is sometimes convenient to define a variance structure as the sum of two simpler terms. Then, it is necessary to give the same variance model to each term and use parameter constraints to equate the parameters. If there are few parameters, this can be done as follows, where **Line** is coded across families so that **Family** classifies the lines:

```
xfa(dTrial,1).Family 2
5 0 XFA1 !GPFPPF !=%ABCDEFGH
0.72631 0.000 .242713 0.000
.882465 .846305 .04419 .743393
Family 0 GIV1
```

```
xfa(dTrial,1).Line 2
5 0 XFA1 !GPFPPF !=%ABCDEFGH
0.72631 0.000 .242713 0.000
.882465 .846305 .04419 .743393
Line 0 GIV2
```

However, for a larger term, there may not be enough letters in the alphabet and so !VCC is required as in:

```
!VCC 1
...
xfa(dTrial,1).Family 2
5 0 XFA1 !GPFPPF
0.72631 0.000 .242713 0.000
.882465 .846305 .04419 .743393
Family 0 GIV1

xfa(dTrial,1).Line 2
5 0 XFA1 !GPFPPF
0.72631 0.000 .242713 0.000
.882465 .846305 .04419 .743393
Line 0 GIV2
21 29 !BLOCKSIZE 8 # The 8 parameters 29:36 are constrained
# pairwise to equal the 8 parameters 21:28
```

Another method actually defines the two matrices as the same matrix (reducing the number of parameters) rather than just equating the parameters. It does this by naming the structure:

```
xfa(dTrial,1).Family 2
5 0 XFA1 !GPFPPF !NAME 'FIVE'
0.72631 0.000 .242713 0.000
.882465 .846305 .04419 .743393
Family 0 GIV1

xfa(dTrial,1).Line 2
!USE 'FIVE' # Structure and parameter values as given above
Line 0 GIV2
```

which associates the model definition labeled FIVE with the second structure.

2.12 PREDICT extension

Associated factors

`!ASSOCIATE factors` facilitates prediction when the levels of one factor group or classify the levels of another, especially when there are many levels. `factors` is an list of factors in the model which have this hierarchical relationship. Typical examples are individually named lines grouped into families, usually with unequal numbers of lines per family, or trials conducted at locations within regions.

Declaring factors as associated allows ASReml to combine the levels of the factors appropriately. For example, in the preceding example, when predicting a trial mean, to add the effect of the location and region where the trial was conducted. When identifying which levels are associated, ASReml checks that the association is strictly hierarchal, tree-like. That is, each trial is associated with one location and each location is associated with only one region. If a level code is missing for one component, it must be missing for all.

Averaging of associated factors will generally give differing results depending on the order in which the averaging is performed. We explore this with the following extended example. Consider the mean yields from 15 trials classified by region and location in Table 2.2.

Table 2.1 Trials classified by region and location

Region	location							
	L1	L2	L3	L4	L5	L6	L7	L8
R1	T1, T2	T3, T4, T5	T6					
R2				T7, T8	T9, T10, T11	T12, T13	T14	T15

Table 2.2 Trial means

T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11	T12	T13	T14	T15
10	12	11	12	13	13	11	13	11	12	13	10	12	10	10

Assuming a simplified linear model `yield ~ mu region location trial` the predict statement `predict trial !ASSOCIATE region location trial` will reconstruct the 15 trial means from the fitted mu, region, location and trial effects.

Given these trial means, it is fairly natural to form location means by averaging the trials in each location to get the location means in Table 2.3.

Table 2.3 Location means

L1	L2	L3	L4	L5	L6	L7	L8
11	12	13	12	12	11	10	10

These are given by

`predict location !ASSOCIATE region location trial !ASAVERAGE trial`
or equivalently

`predict location !ASSOCIATE region location trial`

since the default is to average the base associate factor (trial) within the associated classify factor (location).

By contrast, by specifying

`predict location`

or equivalently

`predict location !AVERAGE region !AVERAGE trial`

ASReml would add the average of all the trial effects and the average of the region effects into all of the location means which is not appropriate. With `!ASSOCIATE`, it knows which trials to average (and which region effects include) to form each location mean. That is, ASReml knows how to construct the trial means including the appropriate region and location effects, and which trials means to then average to form the location table.

However, for region means, we have a choice. We can average the trial means in Table 2.2 according to region obtaining region means of 11.83 and 11.33, or we can average the location means in Table 2.3 to get region means of 12 and 11.

The former is the default in ASReml produced by

`predict region !ASSOCIATE region location trial !ASAVERAGE trial`

or equivalently by

`predict region !ASSOCIATE region location trial`

Again, this is *base* averaging.

By contrast,

`predict region !ASSOC region location trial !ASAVE location trial`

(or `predict region !ASSOC region location trial !ASAVE location`)

produces sequential averaging giving region means of 12 and 11 respectively.

Similarly, an overall sequential mean of 11.5 is given by

`predict mu !ASSOC region location trial !ASAVE region location`

while `predict mu !ASSOC region location trial !ASAVE region` gives a value of 11.58 being the average of region means 11.83 and 11.33 obtained by averaging trials within regions from Table 2.2, and `predict mu !ASSOCIATE region location trial !ASAVE location` predicts mu as 11.38, the average of the 8 location means in Table 2.3.

Further discussion of associated factors

The user may specify their own weights, using file input if necessary. Thus `predict region ... !ASAVERAGE location {1 2 3}/6 {1 1 1 2 1}/6` would give region predictions of 11.67 and 10.84 respectively derived from the location predictions in Table 2.3. Note that because location is nested in region, the location weights should sum to 1.0 within levels of region when forming region means. The `!AVERAGE (!ASAVERAGE)` qualifier allows the weights to be read from a file which the user can create elsewhere. Thus the code `!ASAVERAGE trial 'Tweight.csv', 2` will read the weights from the second field of file `Tweight.csv`. The user must ensure the weights are in the coding order ASReml uses (`trial` order in this instance, given in the `.sln` file or by using the `TABULATE` command).

It was noted that it is the base `!ASSOCIATE` factor that is formally included in the hyper-table. If the lowest stratum is random, it may be appropriate to ignore it. Omitting it from the `!ASSOCIATE` list will allow it to reenter the Ignore set. Specifying it with the `!IGNORE` qualifier will exclude its effects from the prediction but not ignore the structural information implied by the association.

Normally it is not necessary for any model term to involve more than 1 of the associated factors. One exception is if an interaction is required so that the variance can differ between sections. For example, fitting the terms `at(region).trial` as random effects would allow the trials in region 1 to have a different variance component to those in region 2. Prediction in these cases is more complicated and has only been implemented for this specific case and the analogous `region.trial` case. The associated factors must occur together in this order for the prediction to give correct answers.

The `!ASSOCIATE` effect (with base averaging) can usually be achieved with the `!PRESENT` qualifier except when the factors have many levels so that the product of levels exceeds 2147 000 000; it fails in this case because the `KEY` for identifying the cells present is a simple combination of the levels and is stored as a normal (32bit) integer. However, `!ASSOCIATE` is preferred because it formally checks the association structure as well as allowing sequential averaging.

Two !ASSOCIATE clauses may be specified for example

```
PRED entry !ASSOC family entry !ASSOC reg loc trial !ASAVE reg loc.
```

Only one member of an !ASSOCIATE list may also appear in a !PRESENT list. If one member appears in the classify set, only that member may appear in the !PRESENT list. For example

```
yield ~ region !r region.family entry
PREDICT entry !ASSOCIATE family entry !PRESENT entry region.
```

Association averaging is used to form the cells in the PRESENT table and PRESENT averaging is then applied.

2.13 VPREDICT: PIN file processing

Processing of a .pin file can be activated from within the .as file by including a VPREDICT directive. The VPREDICT line may appear anywhere in the .as file but it is recommended it be placed after the model line. It is recognised by commencing with the characters VPR in character positions 1:3. It is processed after the job has processed. This directive must appear on its own line, commencing in column 1.

- If the .pin file exists and has the same name as the jobname (including any suffix appended by using !RENAME), just specify the VPREDICT directive.
- If the .pin file exists but has a different name to the jobname, specify the VPREDICT directive with the .pin file name as its argument.
- If the .pin file does not exist or must be reformed, a name argument for the file is optional but the !DEFINE qualifier should be set. Then the lines of the .pin file should follow on the next lines, terminated by a blank line.

The only change to the processing has been that the variance component lines from the .asr file are now copied into the .pvc file.

2.14 Iterative Schemes

Behaviour under the !SLOW qualifier has been modified as follows. In the iteration subroutine, if the calculated LogL is more than 1.0 less than the LogL for the previous iteration and !SLOW is set and NIT>1, ASReml immediately moves the variance parameters back towards the previous values and restarts the iteration.

2.15 !CONTINUE

In ASReml 2, using !CONTINUE in conjunction with !RENAME was only effective if the particular .rsv file existed. ASReml 3 writes a *basename.ask* file with the names of .rsv files for the various runs so that !CONTINUE can pick up results from some other run of the job. So now, assuming a top command line of

```
!RENAME !ARG 1 2 // !DOPART $1
```

ASReml will use restart values from PART 1 when it is running PART 2, without the user having to explicitly copy the .rsv file.

XFA extensions

Finding the REML solutions for multifactor Factor Analytic models can be difficult. The first problem is specifying initial values. A second problem is that sometimes the LogL rises to a relatively high value and then drifts away.

One strategy which sometimes works is to hold the previously estimated factor loadings fixed for one round of iterations so that the next factor aims at explaining variation previously incorporated in ψ . Then allow all loadings to be updated for next round.

With multiple factors, some constraints are required to maintain identifiability. Traditionally, this has simply been to set the leading loadings of new factors to zero. Loadings then need to be rotated to orthogonality. Now if no loadings are fixed (i.e. !GP), ASReml will rotate the loadings to orthogonality, and hold the leading loadings of lower factors fixed. They are however updated in the orthogonalization process which occurs at the beginning of each iteration (so the final returned values have not been formally rotated).

In an attempt to make the process easier, these two processes have been linked as an additional meaning for the !ALOADING qualifier. For the first !ALOADING iterations, the loading coefficients for all but the last factor are held fixed. After that, loadings are rotated to orthogonal and updated. If !ALOADING is not set by the user and the model is an upgrade from a lower order XFA, !ALOADING is set to 4.

When using !CONTINUE and progressing XFA(k) to XFA($k + 1$), ASReml3 initialises the next factor at $\sqrt{(\Psi * 0.4)}$, making the loading that is relatively the largest, negative.

The XFA display reported in the .res file has been revised. The current output

- $\mathbf{R}^{-1}\mathbf{e}$ and $\mathbf{R}^{-1}\mathbf{e}/\text{diag}\sqrt{\mathbf{R}^{-1} - \mathbf{R}^{-1}\mathbf{W}\mathbf{C}^{-1}\mathbf{W}'\mathbf{R}^{-1}}$ to the `.yht` file,
- copies lines where the last ratio exceeds 3 in magnitude to the `.res` file
- and reports the number of such lines to the `.asr` file.

Alison is researching ways to determine the appropriate cutoff (currently set at 3 in ASReml). This is definitely work in progress. It is not debugged for multivariate models or XFA models with zero Ψ s, and has not been rigorously validated for all other models.

3 Command file: Merging data files

Introduction

Merge Syntax

Examples

3.1 Introduction

The `MERGE` directive, described in this chapter, is designed to combine information from two files into a third file with a range of qualifiers to accommodate various scenarios. It was developed with assistance from Chandrapal Kailasanathan to replace the `!MERGE` qualifier (see page ??) which had very limited functionality.

The **MERGE directive** is placed **BEFORE** the data filename lines. It is an independent part of the `ASReml` job in the sense that none of the files are necessarily involved in the subsequent analyses performed by the job, and there may be multiple `MERGE` directives. Indeed, the job may just consist of a title line and `MERGE` directives. The **!MERGE qualifier**, on the other hand, combines information from two files into the internal data set which `ASReml` uses for analysis and does not save it to file. It has very limited in functionality.

The files to be merged must conform to the following basic structure:

- the data fields must be TAB, COMMA or SPACE separated,
- there will be one heading line that names the columns in the file,
- the names may not have embedded spaces,
- the number of fields is determined from the number of names,
- missing values are implied by adjacent commas in comma delimited files. Otherwise, they are indicated by NA, * or . as in normal `ASReml` files.
- the merged file will be TAB separated if a `.txt` file, COMMA separated if a `.csv` file and SPACE separated otherwise.

3.2 Merge Syntax

The basic merge command is
`MERGE file1 !WITH file2 !to newfile.`

Typically files to be merged will have common *key* fields. In the basic merge, (`!KEY` not specified) any fields having the same names are taken as the *key* fields and if the files have no fields in common, they are assumed to match on row number. Fields are referenced by name (case sensitive).

The full command is:

```
MERGE file1 [ !KEY keyfields ] [ !KEEP ] [ !SKIP fields ]
```

```
!WITH file2 [ !KEY keyfields ] [ !KEEP ] [ !NODUP ] [ !SKIP fields ]
!TO newfile [ !CHECK ] [ !SORT ].
```

Check output field order *Warning:* Fields in the merged file will be arranged with key fields followed by other fields from the primary file and then fields from the secondary file.

Table 3.1: List of MERGE qualifiers

<i>qualifier</i>	action
!CHECK	requests <code>.asrconfirm</code> that fields having a common name have the same contents. Discrepancies are reported to the <code>.asr</code> file. If there are fields with common names which are not key fields, and !CHECK is omitted, the fields will be assumed different and both versions will be copied.
!KEY <i>keyfields</i>	names the fields which are to be used for matching records in the files. If the fields have the same name in both file headers, they need only be named in association with the primary input file. If the key fields are the only fields with common names, the !KEY qualifier may be omitted altogether. If key fields are not nominated and there are no common field names, the files are interleaved.
!KEEP	instructs <code>.asrto</code> include in the merged file records from the input file which are not matched in the other input file. Missing values are inserted as the values from the other file. Otherwise, unmatched records are discarded. !KEEP may be specified with either or both input files.
!NODUP <i>fields</i>	Typically when a match occurs, the field contents from the second file are combined with the field contents of the first file to produce the merged file. The !NODUP qualifier, which may only be associated with the second file, causes the field contents for the nominated fields from the second file only be inserted once into the merged file. For example, assume we want to merge two files containing data from sheep. The first file has several records per animal containing fleece data from various years. The second file has one record per animal containing birth and weaning weights. Merging with !NODUP <code>bwt wwt</code> will copy these traits only once into the merged file.
!SKIP <i>fields</i>	is used to exclude fields from the merged file. It may be specified with either or both input files.
!SORT	instructs <code>.asrto</code> produce the merged file sorted on the key fields. Otherwise the records are return in the order they appear in the primary file.

The merging algorithm is briefly as follows: The secondary file is read in, *skip* fields being omitted, and the records are sorted on the *key* fields. If sorted output is required, the primary file is also read in and sorted. The primary file (or its sorted form) is then processed line by line and the merged file is produced. Matching of key fields is on a string basis, not a value basis. If there are no key fields, the files are merged by interleaving.

If there are multiple records with the same key, these are severally matched. That is if 3 lines of file 1 match 4 lines of file 2, the merged file will contain all 12 combinations.

3.3 Examples

Key fields have different names

```
!MERGE file1 !Key key1a key1b !WITH file2 !KEY key2a key2b !to newfile
```

Key fields have common name and other fields are also duplicated

```
!MERGE file1 !Key keya keyb !WITH file2 !to newfile !CHECK
```

```
!MERGE file1 !Key key !KEEP !WITH file2 !to newfile
```

will discard records from *file2* that do not match records in *file1* but all records in *file1* are retained.

Omitting fields from the merged file

```
!MERGE file1 !Key key !skip s1a s1b !WITH file2 !skip s2a s2b !to newfile
```

Single insertion merging

```
!MERGE adult.txt !Key ewe !KEEP !WITH birth.txt !KEEP !TO newfile !NODUP  
bwt.
```

Bibliography

- Gilmour, A. R., Anderson, R. D. and Rae, A. L. (1985). The analysis of binomial data by a generalised linear mixed model, *Biometrika* **72**: 593–599.
- Gilmour, A. R., Anderson, R. D. and Rae, A. L. (1987). Variance components on an underlying scale for ordered multiple threshold categorical data using a generalized linear mixed model., *Journal of Animal Breeding and Genetics* **39**: 917–934.

Index

CYCLE extensions, 15

Double Slash (//), 17

mbf(), 12

MERGE, 31

qualifier

- !ALPHA, 7
- !ASSIGN, 3
- !ASSOCIATE in PREDICT, 23
- !AS, 4
- !BLUP, 14
- !CHECK, 32
- !CYCLE, 16
- !DEFINE, 26
- !DIAG, 7
- !DV, 4
- !FGEN, 7
- !FIELD, 13
- !FOLDER, 10
- !FOWN, 10
- !GIV, 7
- !GOFFSET, 8
- !GROUPDF, 9
- !GROUPFACTOR, 12
- !GROUPSDF, 9
- !GROUPS, 8
- !HOLD, 13
- !INBRED, 8
- !JOIN, 16
- !KEEP, 32
- !KEY, 13, 32
- !LAST, 8
- !LONGINTEGER, 7
- !MBF, 12
- !MEUWISSEN, 7
- !MGS, 8
- !NAME, 22
- !NA, 6
- !NODUP, 32
- !NOKEY, 13
- !OUTLIERS, 29
- !QUASS, 7
- !RENAME, 12
- !REPEAT, 7
- !RESIDUALS, 28
- !RFIELD, 13
- !SARGOLZAEI, 7
- !SKIP, 32
- !SORT, 7, 32
- !SPARSE, 12
- !SUBGROUP, 12
- !VPREDICT, 26
- !XLINK, 8

REML, ii

vect(), 15

XFA extension, 27