ASReml 3
Work in Progress

Build: de 1 Apr 2008

A R Gilmour
B R Cullis
R Thompson
NSW Department of Primary Industries, Orange, Australia

Copyright Notice

Contents

1 Implemented changes

Table 1.1 Recent changes and bug fixes

Version	Date	Page	Change/Fix
301de	1 Apr 08	??	!AS qualifier added for diallels Calculation of AOM statistics for random effects has been reworked to overcome problem with AR models. !AILOADING 4 automatically set with XFA model !CONTINUE upgrade.
301dd	28 Mar 08	Revise 9 Bug	Change hpr() to vect() Implement !FGEN Fix format error with !SECTION
301dc	26 Mar 08	Revise	Update to MBF Update to CYCLE
301db	19 Mar	Revise	Update FOWN and pedigree Added Standard Devn of variables to main Data Summary
301da		Review	Update to MBF XFA now works under GLMM
301cz	10 Mar	Review	Update of Threshold model Add !GOFFSET to pedigree groups.

1.1 Command Line options

!VERSION 2

Revised

There have been some changes to the Pedigree qualifiers. Use the pedigree line qualifier `!OLD` (or the command line option `V2`, or the top command line qualifier `!VERSION 2`) to have the pedigree qualifiers operate as defined for ASReml 2.

Table 1.2 June 2007 changes and bug fixes

Version	Date	Page	Change/Fix
301cy	4 Feb	Bug	MBF interface bug fix
		30	Predict NESTing updated.
301cw	2 Feb 08		`!D`v does `!D* !Dv`v Quiet mode: open *basename*`.was`, delete on exit. Add `!PNG` (`!GRAPH 21, !HARD 21`) graphics type. Extend `!FILTER` f `!SELECT` v to allow v to be a quoted string in TABULATE and Data Line. Bugfix: predicting average over !GROUP variable. Bugfix: `!VCC 1` input `56:105` 106:155 now defines 50 constraints. New: `!NAME` and `!USE` qualifiers for variance models.(29)
301cv	24 Jan 08		Recognise `at(`F, i`)` as a dependent variable Allow `mu` to be different from `Trait` and `lin(Trait)`
301cu	23 Jan 08	Bug	Initialized AddressY in aiprd (DB/RVV)
301ct	17 Dec 07	34	XFA processing changes
		21	CYCLE changes
		20	MBF changes
301cr	7 Dec 07	Bugs	Threshold model - ensure thresholds in order. Bug in .ask file extension Bugs in !MERGE !NODUP processing

Table 1.3 June 2007 changes and bug fixes

Version	Date	Page	Change/Fix
301cq	29 Nov	20	Further `mbf()` extension. Further `!CYCLE` extension.
301cp	21 Nov	Bug	ln file level labelling: fixed present table overflow: catch overflow Rinv addressing error : fixed Threshold model bounds bug fixed
		30	Added documentation for `!NEST` in PREDICT
301co	21 Nov	Bug	DB compiler bug in aiscor !I level processing bug (GL) xfa !CONTINUE bug model parsing bug
301cm	7 Oct	Bug	`!LAST` interaction with `and()` fixed Improved Lagrangian equation ordering Updated MERGE directive.
301cl	23 Oct 07	Bugs	Fixed problem in ordering routine of slow ordering when covariates included in SPARSE. Fixed absorption problem which gave covariances without variances in C matrix during absorption when genetic groups.
301ck	19 Oct 07	Bug	Fixed gamma update bug (Simon)
301cj	19 Sept 07	Bug	Fixed Intel Pedigree reading bug Fixed AOM/MET bug Adjusted PREDICT workspace calculation.
		Bug	Revised handling of !ADJWTS in multivariate weighted analysis.
301ce	31 August	36	AOM residuals + !MERGE
301cc	01 August	17	`!GROUP` is used to define a factor derived by merging levels of another factor.
		36	A `.vll` output file holds factor level labels for binary data file analysis
		36	`!OUTLIERS` on the data line performs AOM calculations for random effects and residuals.

Table 1.4 June 2007 changes and bug fixes

Version	Date	Page	Change/Fix
		Revision	If a pedigree is not sortable, ASReml 3 now reports the pedigree chain causing the problem
		Revision	`!FOWN` statements may now appear immediately after the model line. A bug causing `mu` to be ignored when `!FOWN` preceeds the model, is fixed.
		Bug	Negative heritabilities: wrong value reported.
301cb	18 July	23	A double slash may be specified to mark a new line in the `.as` file.
		Bug	`xfa()` may now use a factor created using `!SUBSET` as its argument.
		Revision	When a pedigree is processed, the numbers of generations, sires, dams and grandparents are reported.
301ca	12 July	Bug	XFA with Zero PSI failed if hidden factors listed after XFA terms
		Bug	Miss set levels of !A factor if after !G factor.
		18	!HOLD n holds parameters 1..n fixed.
301ca	6 July	7	Extend `NA` *v* to allow *v* to be another field.
301bz	27 June	33	`.ask` file tracks latest run for each trait.
		15	`!FOWN` allows user to specify F-con tests
301by	22 June	Bug	Fix predict plot bug
		Bug	Fix AIDTIN bug introduced with !FOLDER (bx)
			Allow SCREEN involving interactions with Warning
301bx	18 June	Bug	When resetting workspace, write to .asr first in case reset fails.
		Bug	Labelling of Stratum Variance table
		Bug	Fix `!PIN` bug associated with cycling through multiple arguments; check `!JOIN` also
		15	`!FOLDER` *path* sets PATH to data files when the data is not in the working directory.
301bw	14 June	20	Extend `!MBF` to include `!SPARSE` option
		Bug	Change GAMMA to RGMSTR in DO 510 loop of G5VVSF

Review
1. $\surd$ Deprecate
2. $\surd$ If must have, put onto Pedigree line $\surd$ and use different name.

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

1.3 Factor definition

!AS declaration

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 ith 'male' is the same individual as the ith female. If they are not directly coded that way, a pedigree file can be used, but the new `!AS` qualifier is more convenient. It is used as in the following example:

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

and works internally by using a common list of factor level labels for the two factors.

1.4 Transformations

Drop transformation

Revised

- The `!D` transformation conditionally 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 the test field is missing. A new form `!DV` has been implemented which does not drop the the record when the test field is missing 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 $< v$,
  133`!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 $> v$.

Review
1. √ Robin unhappy that `!D` has a changed action. Suggests new form has a new name `!DV` (Drop value), but then need to reinstate old form or internally expand `!D` v to `!D * !D` v.
2. √ Resolve ambiguity I J K !I !+K etc.
UserGuide was wrong to imply `!I` might change focus to variable `I`. Now `!TARGET I` will do so. One problem is that qualifiers are converted to upper case hence need for space in `!TARGET` form.
3. √ tested in az06.as
4. √ Other examples Robin identified: Use of `!TARGET` resolves them
`!Trrr` — is interpreted as `!Time`
`p60 !=trrr !p61 !=trrr !+1 # PEDIGREE`
`p60 !=trrr !a61 !=trrr !+1 # ALPHA!!` These are all invalid coding and will be misinterpreted but not flagged as wrong.

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.

1.5 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 when the usual additive (inverse) relationship matrix is formed.

Review
1. √ `!METHOD 2` is Meuwissen or?
2. √ Build `!FGEN` into MEUWISSAN method.
3. √ Drop old Verbyla routine.
4. √ Leave new Oakey/Verbyla as separate program for time being.
5. √ Spelling of SARGOLZAEI

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 modeified.

- 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 $\boldsymbol{A}^{-1}$ matrix each run. Otherwise ASReml will retrieve the inverse from the `ainverse.bin` file if it is present.
`!Method` *m*	selects method (see below)
`!QUASS`	is `!METHOD 1`
`!MEUWISSEN`	is `!METHOD 3`
`!SARGOLZAEI`	is `!METHOD 4`
`!OLD`	selects ASReml 2 procedures
`!REPEAT`	allows repeat identifiers
`!SORT`	sorts the pedigree into birth order and writes out to a new file.

- Those relating to output

`!DIAG`	writes names, inbreeding, diag($\boldsymbol{A}^{-1}$) *basename*`.aif` (formerly written to `AINVERSE.DIA`).
`!GIV`	writes $\boldsymbol{A}^{-1}$ to *basename*`_A.giv` (formerly written to `AINVERSE.GIV`) in GIV format.

Revised

- 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 $\boldsymbol{A}^{-1}$ 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 $\boldsymbol{A}$ matrix with random effects which at the very least is confusing. When the $\boldsymbol{A}$ 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` f	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 the accessible as `GIV1` or as specified in the output.

Different options are available with different methods for $\boldsymbol{A}^{-1}$, and even if two options apply to the same method, they are not necessarily both allowed together.

Routine	QUASS	MEUWISSEN	SARGOLZAEI
Method	1	2/3	4
`!FGEN`		$\surd$	
`!INBRED`		$\surd$	
`!MGS`		$\surd$	
`!SELF`		$\surd$	
`!GROUPS`		$\surd$	
`!Goffset`		$\surd$	

Oakey et al report an analysis where they fit a factor analytic (XFA) to Clone effects across sites. A subset of their code is reported showing how this nodel is fitted with the new facilities.

```
!WORK 700 !LOG
MET with Dominance
 Subtrial !A
 Trial 6 !A  row 58     column 30
 block 2     tch        ccs
 Clone !P !LL 26
 Family !A   lrow       lcol
 fam !A      familyn    familyas 187
 line 48
 famlin 2267 !A !SORT
 dTrial 4 !A
 iTrial 2 !A

!ASSIGN  BETT fam(Clone)
!ASSIGN  WINT Clone
```

```
CAT99_FAT03SN.ped  !skip 1 !GIV !ALPHA !DOM 2

adi.asd !skip 1 !mvinclude !maxit 24 #!AISING

TAB ccs ~ fam(Clone)  !count  # Identifies 193 families with data

ccs~-1 Trial,
at(Subtrial,1).lcol, #bin03-31
at(Subtrial,1).lrow, #bin03-31
at(Subtrial,2).lrow, # bin03-32
!r Subtrial  1.90860  xfa(Trial,1).Clone iTrial.ide(Clone),

xfa(dTrial,1).$BETT, xfa(dTrial,1).$WINT ,

 Subtrial.block 0.01,
at(Subtrial,3).row 0.152, # bin99-21
at(Subtrial,11).row  0.146057, #  mybos-32
at(Subtrial,3).column 0.036, # bin99-21
at(Subtrial,9).column 0.11, # mnq99-22
!f mv
11 2 4 !NODISPLAY # number of sites # number of R-str # G-str
14 column AR 0.59 !S2=2.86 # 1
8 row    AR 0.50
14 column AR 0.168 !S2=1.445 # 2
8 row    AR -0.018
30 column AR 0.07125 !S2=1.36 # 3
46 row    AR 0.0819
16 column AR 0.439 !S2=0.421 # 4
7 row    AR 0.246
16 column AR 0.104 !S2=0.474 # 5
7 row    AR 0.201
14 column AR -0.2217 !S2=0.4205 # 6
8 row    AR 0.01
14 column AR 0.0814 !S2=0.311 # 7
8 row    AR -0.045
16 column AR 0.29 !S2=0.22 # 8
58 row    AR 0.25
8 column AR 0.16 !S2=1.04 # 9
27 row    AR 0.103
16 column AR 0.24 !S2=0.55 # 10
7 row    AR -.01
```

```
16 column AR -.02 !S2=0.51 # 11
7 row    AR 0.01

xfa(Trial,1).Clone 2
7 0 XFA1 !GFFP
.000 .000 .16 .28 .17 .67
1.94 .99 .10 .94 .55 .71
Clone 0 AINV

iTrial.ide(Clone) 2
2 0 CORGH !GP !+3
.4 .1 .1
ide(Clone) 0 ID

xfa(dTrial,1).$BETT 2
5 0 XFA1 !GPFPFP !=%ABCDEFGH
0.72631 0.000 .242713 0.000
.882465 .846305 .04419 .743393

$BETT 0 GIV1

xfa(dTrial,1).$WINT 2
5 0 XFA1 !GPFPFP !=%ABCDEFGH
0.72631 0.000 .242713 0.000
.882465 .846305 .04419 .743393

$WINT 0 GIV2
```

The `!DOM 2` qualifier defines GIV1 as the between family dominace component and GIV2 as the within family dominance component. It also create as mapping of Clone to Familiy which is accessed by using the fam(Clone) model term.

This 2 part formulation is much more efficient from a procession perspective than using the full dominance matrix directly since $\boldsymbol{D} = \boldsymbol{Z}_F \boldsymbol{D}_{BF} \boldsymbol{Z}_F' + \boldsymbol{D}_{WF}$ is dense of order 2663 but $\boldsymbol{D}_{BF}$ is only of order 410. In fact, Oakey et al used a smaller matrix still because only 193 families in the pedigree are directly represented in the data. However, the matrices are computed from the pedigree without knowing what is actually in the data. Nevertheless, using 410 families makes the analysis feasible.

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 `!GROUPDF` 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 `!GROUPS 7` begins

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

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

Review
1. √ Consider generalizing to add diagonal as alternative to adding these constraints.
2. √ Need to define these anyway: see `!GROUPS`
3. √ Explain importance of issue: see `!GROUPS`
4. Mechanism for zeroing equations as alternative. ARG comments: I once started defining a !ZERO qualifier and then developed !LAST as an alternative. A possible syntax is `!ZERO` *model-term list*. Implementation would then be to declare these as 'singularities' before absorbing the mixed model equations.
5. √ Using `!LAST` may be sufficient.

1.6 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
```

Review
1. √ Check it also applies to pedigree/giv files. Applies to pedigree, giv, data and include files (test: met.as).
1.a What about say mbf and other input files!
2. May need mechanism to set folder for SCRATCH files

!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

Revised

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.

Revised

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

F(`A ; mu B C`),
F(`B ; mu A C`),
F(`C ; mu A B`),
F(`A.B ; mu A B C B.C A.C`),
F(`B.C ; mu A B C A.B A.C`),
F(`A.C ; mu A B C A.B B.C`) and
F(`A.B.C ; mu A B C A.B A.C B.C`).

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 `O`, `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.

Review
1. Robin and Brian think some users will want multiple tests of some terms. As this would involve a complete restructure of the ANOVA table, Arthur is reluctant to pursue this.
2. Re-word so 'maximal model' is clearly defined.
3. $\surd$ Replace `|` with another character (+) to avoid confusion as this use of `|` conflicts with the meaning given in the following lines. Have used `;` to replace `|` as `+` is already a valid operator in a model line.
4. $\surd$ Notify of extra terms rather than just ignore.
5. $\surd$ FOWN replaces Fcon; does it work for alg and num derivatives? YES does it have maximal/conditional model issues? SAME as F-CON.
6. $\surd$ When will it do proper Kenward tests? Wald statistics are currently calculated from an unadjusted variance matrix, $\mathbf{\Phi}$. The User Guide says *Kenward and Roger (1997) pursued the concept of construction of Wald-type test statistics through an adjusted variance matrix $\mathbf{\Phi}_A$. They argued that it was useful to consider an improved estimator of the variance matrix $\mathbf{\Phi}_A$ which has less bias and accounts for the variability in estimation of variance parameters. Firstly the small sample distribution of Wald statistics is simplified when the adjusted variance matrix is used, Secondly, if measures of precision are required for $\mathbf{\Phi}_A$, or effects therein, those obtained from the adjusted variance matrix will generally be preferred.*
ASReml has `!KADJ 1/2` option to calculate $\mathbf{\Phi}_A$ algebraically or by absorption of the working variables, for use in calculating F statistics (F-inc and F-con). However, the resultant values are not always what Arthur expected so the option has not been promulgated until someone can validate the calculations.

!GROUP

The `!GROUP` 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
`!GROUP` *<Group_factor> <Exist_factor> <new codes>*
for example
`!GROUP Year YearLoc 1 1 1 2 2 3 3 3 4 4`
forms a new factor `Expt` with 4 levels from the existing factor `YearLoc` with 10 levels.

Revised Notice that for default averaging in prediction, the weights for the levels of the grouped factor (`Year`) will be (in this example 0.3 0.2 0.3 0.2) derived from the weights for the base factor (`YearLoc`). Use `!AVE YearLoc { 2 2 2 3 3 2 2 2`

`3 3 }/24` to produce equal weighting of `Year` effects. Incomplete mapping of one to the other will usually lead to prediction problems.

Review
1. Review predict comments. If Year and not YearLoc is fitted, default averaging would be expected to be equal weights!

!HOLD

Revised

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. E.g. `!HOLD 1:20 30:40`.

Review
1. √ Extend to a list e.g. `!HOLD 1:20 30:40`. [met.as]

1.7 Model Terms

Family

`fam(`*Clone*`[,`*field*`])` allows recoding of a factor when the factor is large by supplying the recoding information in a separate file. The term must be defined with a
`!FAM` *term filename*
line supplied between the data line and the model line.

The motivating example was the analysis of data on Clones. For efficient fitting of a dominance variance matrix, the family covariance is supplied as a family matrix. A simplified example of the coding follows.

```
 ...
 Clone !P
 ...
Clone.Ped
DBF.GIV
DWF.GIV
```

```
Clone.data
!FAM fam(Clone) Family.txt # maps Family to Clone in Pedigree order
#    DBF.GIV, DWF.GIV and Family.txt all derived from Clone.ped
 yield ~ mu ... !r Clone giv(fam(Clone),1) giv(Clone,2)
```

Review
Robin noted that `!FAM` is an alternate form of `!GROUP`, `!MBF` is an alternate form of `!CONTRAST` so this may need rationalization, or renaming.
There are seven related forms:

Qualifier	Form	Inter Code
`!CONTRAST`	*TermName BaseFactor List*	-25
`!FAMILY`	`fam(`*BaseFactor,Column*`)` *Filename*	-33
`!GROUP`	*TermName BaseFactor List*	-37
`!MBF`	`mbf(`*BaseFactor,Columns*`)` *Filename*	-24
`!SUBSET`	*TermName BaseFactor List*	-27
`!SPLINE`	`spl(`*BaseFactor,Degree*`)` *List*	-3

Use of `!MBF` to process markers is an extension which doesn't look like basis functions, and `!RENAME` was introduced to generalise it.
So could combine `!CONTRAST` and `!MBF` by adding `!IN` *filename*,
`!GROUP` and `!FAMILY` by adding `!IN` *filename*.

vec covariate

`vect(`*TVC*`)` (vec transpose) is added to allow multivariate data presentation with a trait specific covariate. The test example included

```
 channel 2
 signal !G 93  # 93 slides
 background !G 93
dart.asd  !ASUV
 signal ~ Trait.channel channel.vect(back) ...
```

to fit a slide specific regression of `signal` on `background`. In this example, `signal` is a multivariate set of 93 variates and `back` 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 back`. 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.channel slide.channel.back` .

Review
1. $\surd$ Robin suggests `vct()` (standing for vec transpose) rather than `hpr()` and `Trait` should be implicit.
2. $\surd$ Better explanation of multivariate model in UG.

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 specifiy 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*
Revised
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/marker number are to come from a substitution variable since they are different numbers. If the *key* field just contains the numbers 1:*n* in order, 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.

Review
1 $\surd$ Propose `!NOKEY` implying `key` is $1:n$
2 χ FAM Can we have fac(mbf(marker,1)) to avoid the need for fam ?
3 $\surd$ MyBasis options: Hate !FROM if cannot have !SKIP why not !CSKIP
34 the !FROM is presumably !FROM 36
$\surd$ Changed to `!FIELD`

1.8 !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. 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`

Revised

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.

Revised

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 it is still also generated by the `!CYCLE` unless it is modified to read
`!CYCLE 1:34 36:1000` .

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
```

Review
1. $\surd$ What about `$J`? Now defined by CYCLE.
2. $\surd$ Check what happens if no `!JOIN` with and without `!RENAME`. Maybe make !JOIN the default.
`!JOIN` is now the default.

3. $\surd$ Added explanation. Example
... fits 1000 jobs fitting successively the 1000 Marker covariables defined in the 1000 data files. Subsequent use of
`grep LogL: cycle.asr | sort > cycle.sum`
gives a sorted summary of the jobs.
4. $\surd$ Add `!STEP` *s* option and allow substitution strings `$I+`*i* to pick up CYCLE arguments in sets.
5. Describe use of script files for multiple runs.

1.9 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
```

1.10 Multivariate data presentation

ASReml will only allow up to 20 dependent terms to be nominated but these may be grouped `!G`n sets of variables so that more than 20 variates may be analysed (usually in conjunction with !ASUV). (See the preceding example on page 19).

Review
1 √ What transformations apply to `!G` sets of variables? `!RESCALE` o s is the only general transformation that covers a group of variables. It adds o (an offset) and multiplies by s (a scaling factor).
2. √ What's the difference between variables and variates? They are synonyms to me!

1.11 Multiple Threshold Models

ASReml 3.01 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 !THR 3 ~ Trait variety !r rep   # Alternative form
at(lodging,1) at(lodging,2) at(lodging,3) !THR 3 ~  Trait variety,
!r rep

PREDICT variety
```

In this coding, 8 varieties are dropped from the analysis since no lodging occurred in these varieties on any plot, putting them in a extreme class, not well modelled as fixed effects on an underlying scale (logit) scale.

The threshold model is internally fitted as a cumulative multivariate model. The 'traits' analysed should be counts in the ordered categories (omitting the last category). These are summed internally to form the cumulative distribution which is modelled as logit (`!LOGIT`), probit (`!PROBIT`) or Complementary LogLog (`!CLOG`) variables. The `!THR` t qualifier requesting a threshold analysis is required and specifies the number of thresholds (should equal the number of traits as 'threshold+normal' not yet implemented), `!TOTAL` n indicates the variable containing totals. With ungrouped data (above), `!TOTAL` n is not required (being implicitly 1), and the `at(.,.)` model term can be used to convert the scores into the separate 'traits'.Alternatively, the score variable alone may be specified provided scores $1 \cdots t$ are al represented in the data. Revised

Predicted values are reported for the cumulative proportions.

The next example is the analysis of grouped data for three ordered classes (5, 4, 3) where `Total` is the number in the group, `Score5` is the number which scored 5, `Score4` is the number which scored 4 and `Score3`, calculated by difference, is the number remaining.

The ASReml coding to fit a two threshold model to the Score5/Score4 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
  Score5  Score4  Scald  Rot
  Score321 !=Total !-Score4 !-Score5
lamb.dat !skip 1  !DDF -1

Score5 Score4 !thr 2 !TOTAL=Total ~ 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 heterogenity factor [Deviance/DF]     0.97

        - - - Results from analysis of Score5 Score4 - - -
Notice: While convergence of the LogL value indicates that the model
        has stablized, 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 hypertable 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.    Score5    1.4511         0.1952 E           0.8102         0.0318
    1.    Score4    3.7217         0.2114 E           0.9764         0.0054
    2.    Score5    0.7239         0.1915 E           0.6735         0.0434
    2.    Score4    2.9945         0.2064 E           0.9523         0.0103
    3.    Score5   -0.3138         0.2986 E           0.4222         0.0707
    3.    Score4    1.9567         0.3048 E           0.8762         0.0370
    4.    Score5    0.2571         0.1913 E           0.5639         0.0475
    4.    Score4    2.5277         0.2043 E           0.9261         0.0153
    5.    Score5    0.5355         0.1444 E           0.6308         0.0342
    5.    Score4    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 Score 5 for Group 1, 0.9764 is the proportion in Score 5 and 4.

Review

1. $\surd$ Suppress printing of Identity US R structure.
2. $\surd$ Allow `at(lodg,1) at(lodg,2) at(lodg,3) !THR 3` to be specified as `lodging !THR`. Note that this only works if `lodging` is NOT defined as a factor. If it is a factor with 9 levels, ASReml will set up 9 binary variables as the dependent variable, one too many.
3. $\surd$ Allow `!GROUP Glodge lodging 1 2 0 0 // Glodge !Thr 2 ...`; Note that the top class needs to be assigned the value 0 (`1 2 3 3` will generate an error.)
4. $\surd$ Add explanation that 'LogL' is not suitable for testing in GL(M)M models. Report deviances each iteration.
5. χ Add option to return class numbers rather than just cumulative values from PREDICT.
6. $\surd$ Review issue from page 47 esp why AISING: AISING need no longer be explicitly specified.
7. $\surd$ New dataline qualifier `!GLMM` i sets the number of inner loops (default and minimum is 2) performed in each outer GLMM iteration.

1.12 Nested R structure

The nested R structure is implemented in two forms.

Nested AR

The test example (pbid) had a series of bids for several auctions. There were unequal numbers of bids. The model included a spline to fit the overall trend, and random linear regression as a base for comparing auctions. The R structure was AR assuming data in bidding order withing auctions.

Revised

The syntax for this R structure is
```
1 1
0 0 AR1 .5 !subsection auction
```

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

Review
1. $\surd$ Improve description. $\boldsymbol{R} = \oplus_{i=1}^{s} \boldsymbol{R}_i$ where $\boldsymbol{R}_i = \otimes_j^c = 1\Sigma(\phi_{ij}$ and $\Sigma(\phi_{i1}$ mayn have direct sum structure with common parameters.
2. $\surd$ Change `!NEST` to `!SUBSECTION`
3. χ Extend at least to US structure

Nested correlations based on explicit times

The motivation was Sue's problem sue/r21c.as which had 80 `EXP` structures of varying size based on DATE. Here the syntax is
```
1 1 0
0 date EXP .256 !subsection plot
```
indicating that the levels needed for calculating the `EXP` correlations are in the variable date. NB This form is available for all the 'Kriging' type models.

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:

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

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

Revised

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).$BETT 2
5 0 XFA1 !GPFPFP
0.72631 0.000 .242713 0.000
.882465 .846305 .04419 .743393
$BETT 0 GIV1

xfa(dTrial,1).$WINT 2
5 0 XFA1 !GPFPFP
0.72631 0.000 .242713 0.000
.882465 .846305 .04419 .743393
$WINT 0 GIV2
 21:28 29:37
```

Revised

Another method under evaluation is

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

xfa(dTrial,1).$WINT 2
```

```
5 0 XFA1 !GPFPFP !USE 'FIVE'
# Initial parameters now omitted here
$WINT 0 GIV2
```

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

1.13 PREDICT extension

Nested factors

Revised

`!NEST` <*factors*> facilitates prediction when the levels of one factor are nested in the levels of another. <*factors*> is an list of nested factors, ordered with latter factors nested within earlier ones. Typical examples are say 1000 individually named lines which represent 100 families, or 100 experiments representing 20 locations in five regions.

When averaging, it can be at several levels. For example, given

```
region     1  1  1  1  1  1  2  2  2  2  2  2  2  2  2
location   1  1  2  2  2  3  4  4  5  5  5  6  6  7  8
experiment 1  2  3  4  5  6  7  8  9 10 11 12 13 14 15
```

`predict experiment !nest region location experiment` includes the appropriate levels of `region` and `location` in the prediction of experiment.
`predict location !nest region location experiment`
adds in the associated `experiment` effects with weights
`1/2 1/2 1/3 1/3 1/3 1/1 1/2 1/2 1/3 1/3 1/3 1/2 1/2 1/1 1/1`.

However, averaging over several nested factors requires careful consideration. Consider the following nested structure, and consider what is the overall mean. Averaging can occur two ways.

	A1	A2	B1	B2	B3	C1	C2	C3	C4	Mean
A	10	12								11
B			9	10	11					10
C						7	8	9	12	9
Mean	10	12	9	10	11	7	8	9	12	9.77 or 10

So, if we are averaging over nested factors, we need to consider the order of averaging. The default is to average at the lowest level, giving a mean in the example of 9.77.

`predict region !nest region location experiment` will produce `region` means which are the averages of experiments in the region. Thus, averaging at the `experiment` level, the `location` weights would be `2/6 3/6 1/6 0 0 0 0 0` and `0 0 0 2/9 3/9 2/9 1/9 1/9` and the `experiment` weights `1/6 1/6 1/6 1/6 1/6 1/6 0 0 0 0 0 0 0 0 0` and `0 0 0 0 0 0 1/9 1/9 1/9 1/9 1/9 1/9 1/9 1/9 1/9`.

`predict region !nest region location experiment !AVE location` will produce averaging at the `location` level and use `location` weights `1/3 1/3 1/3 0 0 0 0 0` and `0 0 0 1/5 1/5 1/5 1/5 1/5` and `experiment` weights of `1/6 1/6 1/9 1/9 1/9 1/3 0 0 0 0 0 0 0 0 0` and `0 0 0 0 0 0 1/10 1/10 1/15 1/15 1/15 1/10 1/10 1/5 1/5`.

Of course, another option is to ignore the lower stratum if they are random probably by simply not including it in the `!NEST` list (if it is RANDOM) or explicitly with `!IGNORE experiment`.

`!NEST` formally checks the coding is hierarchal. If a level code is missing for one component, it must be missing for all.

The `!NEST` effect 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, `!NEST` formally checks that there is a nested structure as well as allowing averaging at a higher level.

Only one `!NEST` qualifier is allowed but it can be used together with a `!PRESENT` qualifier as in

```
yield ~ region !r region.family entry
PREDICT entry !NEST family entry !present entry region
```

More complicated nest averaging: Consider 2 states containing 2 and 3 regions, containing 2, 3, 2, 3, 2 locations with 3, 2, 3, 2, 1, 3, 2, 3, 2, 1, 3, 2 experiments (total 27). Averaging everything at the experiment level gives state weights of (11, 16)/27, region weights of (5, 6, 5, 6, 5)/27, location weights of (3, 2, 3, 2, 1, 3, 2, 3, 2, 1, 3, 2)/27 and equal experiment weights of 1/27.

Now, if we wanted to give average weights at the Location level, the required weights are: for state (5, 7)/12, for region (2, 3, 2, 3, 2)/12, for location (1/12) and for experiment values of 1/36 1/36 1/36 1/24 1/24 $\cdots$ or 1/12 depending on whether there are 3, 2 or 1 experiments at the location.

Now, if we wanted to give average weights at the Region level, the required weights are: for state (2, 3)/5, for region 1/5 but there are two ways of weighting location and experiment. Averaging averages gives for location (1/10 1/10 1/15 1/15 1/15 1/10 1/10 1/15 1/15 1/15 1/10 1/10) and for experiment values of 1/30 1/30 1/30 1/20 1/20, 1/45 etc depending on whether there are 3, 2 or 1 experiments at the location. Averaging just by experiment gives for location (3/25 2/25 3/30 2/30 1/30 3/25 2/25 3/30 2/30 1/30 3/25 2/25) and at the experiment level (1/25 ... 1/30 ... 1/25 ... 1/30 ...1/25). The former should be specified as `!AVE Region !AVE Loc`.

Review
1. $\surd$ The preceding description is revised to reflect the new fomulation of !NEST.
2. $\surd$ Number of strata being increased.
3. Review PRWTS example. Present as table to make clearer.
4. $\surd$ Including a term in !NEST puts it in the hypertable so that it is either included in the prediction or averaged.
5. $\surd$ Revise 'predict description' in .pvs so more obvious what is going on.
6. Revise description in User Guide.
7. χ The `!GROUP` model term redefinition also defines nesting but there is currently no way to average according to grouped levels: see TC3
8. $\surd$ Slight speed up of PREDICT calculation with an indexed list.
9. $\surd$ Test example `srleg.as`

1.14 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.

In the future

Provide a better general specification of components.

1.15 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.

Review
Convergence failure occurs when LogL repeatedly decreases `2*NWV+10` and the model is not a GL(M)M and NIT>2. If `!SLOW` is not set, restart when this criterion fails.

1.16 !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 identifiablity. 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.

Revised

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

Following is the coding for a large job tying to estimate four factors.

```
!WORK 1 !NOGRAPH !continue
Title: ALBUS_2stage.
#trial,year,region,variety,yield,rep,weight,ems
#KFA02BURU,2002,NSW,KIEV-MUTANT,0.873,3,2136.562,0.0010000
#KFA02BURU,2002,NSW,ULTRA,0.831,3,2136.562,0.0010000
#KFA02BURU,2002,NSW,KIEV-BC3-A,0.809,3,2219.959,0.0010000
#KFA02BURU,2002,NSW,KIEV-BC3-B,0.853,3,1872.183,0.0010000
 trial   !A
 year    !I
 region  !A
 variety !A
 yield
 rep     *
 weight  !*0.025
 ems
!CYCLE 11 1 2 3 4     !JOIN
```

```
!DOPART $I
ALBUS_2stage.csv  !SKIP 1   !MAXIT 40 !AILOAD 20

!PART 11
 !MAXIT 25
 yield !wt=weight ~ mu trial !r  trial.variety
1 1 1
0 !S2==0.025
trial.variety 2
trial 0 CORUH .1
87*.1
variety

!PART 1 2 3 4
 yield !wt=weight ~ mu trial !r xfa(trial,$I).var
1 1 1
0 !S2==0.025
xfa(trial,$I).var 2
xfa(trial 0 XFA$I     !GP
87*.01
87*.07  87*.07   87*.07  87*.07
variety
```

A previous set of analyses using these five models gave LogL values for the models CORUH, XFA1, XFA2, XFA3 and XFA4 respectively of 2782, 2910, 3021, 3109 and 3200 using the strategies listed above in separate runs. Running this job using the integrated strategy produced LogL values of 2783, 2911, 3048, 3153 and 3206. However, for models XFA3 and XFA4, the LogL drifted away again.

The XFA display reported in the `.res` file has been revised. The current output from a small example with 9 exvironments and 2 factors is

```
DISPLAY of variance partitioning for XFA structure in xfa(Env,2).Geno
Lvl |----+----+----+----+----+----+----+----+----+----| TotalVar %expl PsiVar Loadings
  1 |                                       1         |   0.3339  79.7 0.0679 0.5147 0.0335
  2 |                                               1 2   0.1666 100.0 0.0000 0.4003 0.0797
  3 |                            1    2               |   0.2475  67.8 0.0798 0.3805 0.1514
  4 |                                            1    2   0.1475 100.0 0.0000 0.3625 0.1269
  5 |                                        1        2   0.4496 100.0 0.0000 0.6104 -0.278
  6 |                     1                           2   0.1210 100.0 0.0000 0.2287 0.2622
  7 |                    1     2                      |   0.4106  54.4 0.1872 0.4152 -0.226
  8 |    1                                            2   0.0901 100.0 0.0000 0.0922 0.2857
  9 |                           1                     2   0.1422 100.0 0.0000 0.2819 0.2506
  0 |----+----+----+----+----+----+----+----+-- Average   0.2343  89.1 0.0372 0.3651 0.0763
```

In the figure, `1` indicates the proportion of `TotalVar` explained by the first loading, `2` indicates the proportion explained by first and second (provided it plots right of `1`. Consequently, the distance from `2` to the right margin represents

`PsiVar`. `%expl` reports the percentage of `TotalVar` explained by all loadings. The last row contains column averages.

1.17 Residuals

.vll file

The `!SAVE` qualifier writes a copy of the data (before or after transformation) to a binary file which can be used as input for a later run of ASReml. The `!RESIDUALS` qualifier writes a copy of the data and an extra field containing the residual to a binary file which can be used as input for a later run of ASReml. The difficulty with these options has been that factor labelling was lost, as the binary files just contain the factor levels. Now, when either `!SAVE` or `!RESIDUALS` is specified, ASReml saves the factor level labels to a *basename*`.vll` and attempts to read them back when data input is from a binary file. Note that if the job `basename` is differs between runs, the `.vll` file will need to be copied to the new *basename*. If the `.vll` file does not match the factor structure (i.e. the same factors in the same order), reading the `.vll` file is aborted.

Alternative Outlier Model

The `!OUTLIER` qualifier invokes a partial implementation of research by Alison Smith, Ari Verbyla and Brian Cullis. With this qualifier, ASReml 3 writes

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

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 Ψs, and has not been validated for other models.

Review
1. $\surd$ Method of calculation of divisor for test on random effects has been changed to absorption to fix previous problem with AR models.
2. $\surd$ A problem with the R outlier calculation under certain fixed variance models has been resolved.

1.18 Under Development

This section describes the syntax implemented to test options under development. You are welcome to try the options and give feed back but do not be surprised if they do not work, or only partly work. More detail is contained in ASReml 3 Notes.

Singular G

The present implementation applies to `DIAG`, `US` and `CHOL`n`C` variance structures. It requires doubling the factor size, and is requested with a `!SING` qualifier on the variance structure definition line.

For `DIAG`, singular means some variances are zero. For `US`, singular means some variances and associated covariances are zero. (This is a very restrictive definition of singular.) For `CHOL`k`C`, singular means some conditional variances are zero.

A sample test job follows:

```
Balanced Sire model  Zero components
 row y1 y2 y3 y4 Sire *
bse.asd !skip 1 !DOPART $1 !MAXIT 35 !BRIEF#!EM

y4 y1 y3 y2 ~ Trait !r Sire.xfa(Tr,4)
1 2 1
0
Tr 0 US !GP
10*0

Sire.xfa 2
Sire

!PART 1
```

```
xfa 0 US  !SING  !GPPPZZZPPZP
10*0

!PART 2
xfa 0 US !GU !SING
10*0

!PART 3
xfa 0 CHOL3C !SING
10*0

!PART 4
xfa 0 CHOL2C !SING  !GP
10*0

!PART 5
xfa 0 CHOL1C !SING !GP
10*0

!PART 6
xfa 0 CHOL3C !SING !GPPPPPZPPZP
10*0
```

For experimenting with these models, I have used the `!SUBSET` qualifier to reorder the levels of a factor. For example, if there are 14 levels of `site`, define `Site` as say
`!SUBSET Site site 2 4 6 8 14 12 10 9 7 5 3 1 11 13 14*0`
and then

```
...
 !r Site.geno
...
Site.geno 2
Site 0 Chol4C !SING
...<US initial values in new order>
geno
```

It turns out that the fit of a particular degree (k) of CHOLkC is very dependent on the order.

Simple imputation

Imputation is only partly implemented and not tested. It is invoked by using `!IMPUTE` on the data line, and splitting the model into submodels with the `!SM` *p* qualifier. ASReml will then oscillate between the models.

```
Imputation - fixed model
 row A 100 B 100Y
impute.asd !skip 1   !DOPART $1

!PART 1      A and B fixed
 Y ~ mu  A  B
0
!PART 2      A and B fixed - constrained
 Y ~ mu  c(A)  c(B)
0
!PART 3      A fixed and B random
 Y ~ mu  A !r B
0

!PART 4      A fixed, constrained and B random
 Y ~ mu  c(A) !r  B
0
!PART 5      A and B random
 Y ~ mu !r A  B
0

#   Models with Imputation
!PART 11      A and B fixed
 !IMPUTE
 Y ~ mu !SM 1 A !SM 2 B
0
!PART 12      A and B fixed and constrained
 !IMPUTE
 Y ~ mu !SM 1 c(A) !SM 2 c(B)
0
!PART 13      A fixed and B random
 !IMPUTE
 Y ~ mu !SM 1 A !r !SM 2 B
0
!PART 14    A fixed, constrained and B random
```

```
 !IMPUTE
 Y ~ mu !SM 1 c(A) !r !SM 2 B
0
!PART 15   A and B random
 !IMPUTE
 Y ~ mu !r !SM 1 A !SM 2 B
0
```

Review
Arthur to draft something for UG on SNG and IMP and lets see.

!MERGE directive

At present the is a `!MERGE` directive and a `!MERGE` qualifier in ASReml. This section describes the `!MERGE` directive which has been written in association with Chandrapal Kailasanathan.

The `!MERGE` **directive** is placed BEFORE the data filename lines and invokes a process whereby two distinct files are merged into a third file. 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 typically does not result in a third data file being written. It is very limited in functionality and will be phased out.

Preamble The merge facility is designed to combine information from two files into a third file with a range of qualifiers to accomodate various scenarios. The files however, 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 processing.
- the merged file will be TAB separated if a `.txt` file, COMMA separated if a `.csv` file and SPACE separated otherwise.

The basic merge command is
`!MERGE` *file1* `!WITH` *file2* `!to` *newfile*. Fields are referenced by name (case sensitive).

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. If the files have no fields in common, they are assumed to match on row number.

`!KEY` and `!CHECK`
If the files have common fields to be used for matching, but the names differ, the names can be nominated with the `!KEY` qualifier. For example
`!MERGE` *file1* `!Key` *key1a key1b* `!WITH` *file2* `!KEY` *key2a key2b* `!to` *newfile*.

If the files have common names for the key fields but other fields also have common names, the `!KEY` qualifier need only be listed with the first file name. For example
`!MERGE` *file1* `!Key` *keya keyb* `!WITH` *file2* `!to` *newfile*.

If other fields have common names, they are copied as distict fields unless `!CHECK` is specified. `!CHECK` indicates that other common fields are meant to be the same and only the first version is copied to the output. However, the contents are compared and discrepancies are reported.
`!MERGE` *file1* `!Key` *key1a key1b* `!WITH` *file2* `!to` *newfile* `!CHECK`.

`!KEEP` rows
When lines are matched on key fields, there may be lines that do not have a match in the other file. These lines will normally be excluded from the output file. The `!KEEP` qualifier may be given for one or both of the input files, requesting that all lines from that file appear in the merged file (with `NA` in the fields not filled from the other file). For example
`!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.

`!SKIP` *fields*

It is not necessary that all fields by exported to the merged file. Which fields are exported can be controlled by specifying fields in the input files to be ignored. The `!SKIP` qualifiers are applied on data input. For example
`!MERGE` *file1* `!Key` *key* `!skip` *s1a s1b* `!WITH` *file2* `!skip` *s2a s2b* `!to` *newfile*

`!NODUP` *fields*
When merging 1 to many, different rules may apply for *factor* variables than for data variables. The default action is to use a matching code repeatedly. For example, if the primary file has a `SIRE` field, and you want to associate a `SIREBREED` with the SIRE, the second file may just consist of two fields `SIRE SIREBREED` and the output file will have a `SIREBREED` code in every record. However, in another scenario, you may have a primary data file containing 3 or 4 (annual) fleece measues for a set of animals and the second file relates to the same animals, but has only one record per animal containing say a birth weight. The desired output file will want the birth record to only be copied once (to the first matching record) but not to all matching records.
`!MERGE` *file1* `!Key` *key* `!KEEP !WITH` *file2* `!KEEP !to` *newfile* `!NODUP bwt.`
`!NODUP` should not refer to field which appears in the first file.

`!SORT`
The strategy is to read in the second file and sort it on key fields. The first file is then read and merged line by line. However, if `!SORT` is specified, both files are read in and sorted, and then merged so that the result is sorted on the key fields.

General Note that 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.

Review
Plan:

01	Revise ASReml 3 code in the light of this review,
02	Device test suite of jobs,
03	give code to Dave and Simon for production,
04	review documentation in anticipation of August 2008 release.

Review
Check User Guide:
11 'LogL offset' notes in ASReml/index
12 Unix system notes: use single processor,
13 Unix system: explain working folder uner Unix.
14 explain '.. terms are ignored ..' in predict heading,
15 maybe rephrase to *Model terms involving ... are ignored*
16 √ allow `!select` "*level-name*" here and elsewhere (as in predict)
17 add explanation that ANOVA P values may require !DDF
18 suppress 'FAULT NO' from output
19 check listing of Notices/Warn/Error
20 check spl(dens) in UG for parsing issues.
21 explain relationships of !ALPHA and !P and ordering rules
22 improve error message when pedigree is out of order.
23 explain how to be sure of GIV file order.
24 notify brian and robin when ASReml3 updated on ftp site.
25 advertise ASReml3 forum on ASReml discussion list.
26 Example: Example of Use of MYOWNGDG to model R structure in terms of a small number of parameters

Review
27 Ensure Guide spells out which model functions (eg sin(y)) can be used as dependent variables.
28 Example of use of !TWOSTAGEWEIGHTS
28 Check accessability of Archives.
Wednesday6 and Friday 8:
31 GRM: Generalized Linear Modelling. cf Ari, RSS 1993 Gamma, loglin model on residuals so cycle between calc of weights and fitting model. Weights calculated using loglin model and gamma distribution with residuals as data. Y analysed using weights from GLMM run.
32 `!GENERATE n file` to create a file with n sets of resampled data which can then be analysed and presumably analyses summarised. e.g. PART 1 does GENERATE, PART 2 has cycle to analyse all the samples, picking them up with MBF. After fitting $\boldsymbol{y} \sim \boldsymbol{X}\hat{\boldsymbol{\tau}} + \boldsymbol{Z}\tilde{\boldsymbol{u}} + \tilde{\boldsymbol{e}}$ estimating parameters for $\boldsymbol{R}$ and $\boldsymbol{G}$, use these to resample $\boldsymbol{u}$ and $\boldsymbol{e}$ with which to reconstruct $\boldsymbol{y}$.
33 `!SUMMARY` *keywords* to write selected items to a summary file. What summary is wanted? Could be a summary across CYCLE of particular items, or could be a selection of output.
34 Imputation: Athur would like a worked example to emulate.
35 Simple linear models on variance parameters; use chain rule to collapse large set of working variables to a linear model; fit model parameters, predict original parameters.
36 PREDICT running time; test on large predict where time is taken and whether time is linear on number of predictions. If first columns of D contained $\boldsymbol{W}'\boldsymbol{R}^{-1}\boldsymbol{R}\boldsymbol{Y}^*$, could predict for these extra variables.
37 Own predict design!
38 $\surd$ Bug fixed! with XFA in THRESH models.
39 investigate TC3 XFA drift
40 BINNOR in guide

User Guide notes

re:
I want to do a search about the effects of using incorrect initial values (priors) for variance and covariance estimation. i want to do it from a theoretical study, but every i search i cant find any useful things, as in your software we consider priories in many situation, i thought that may you can help me to find useful references for this.

I am unaware of any formal studies in the REML context. My experience is that it depends on
1) the amount of information in the data on the parameter,
2) the complexity of the model - the scope for other terms to explain variation
3) the smoothness of the likelihood surface (a consequence of 1 and 2 I suppose)
4) the nature of the parameter and its covariance with other parameters (how the model is formulated)
5) the closeness of the parameter (and all parameters) to the REML solution.

There are three possible outcomes:
1) the model converges
2) the model converges but to a local solution
3) the estimation blows up.

Outcome 2) is rare. Usually, if it doesn't converge, it obviously FAILS to converge.

If it fails to converge, the remedial steps are:
1) check the model is plausible given your knowledge of the data.
2) check the initial values are plausible
3) simplify the variance model to get better starting values
4) hold some parameters fixed while estimating others (and cycle through the combinations)

Bibliography

Alwan, M. T. (1983). *Studies on the mating performance of the booroola merino crossbred ram lambs and the foot conditions in booroola merino crossbreds and perendales grazed on hill country*, Master's thesis, Animal Science, Massey University.

Gilmour, A. R. (1983). *The estimation of genetic parameters for categorical traits*, Master's thesis, Animal Science, Massey University.

Gilmour, A. R. (1988). Reg – a generalised linear models program, *Technical report*, NSW Department of Agriculture, Sydney.

Gilmour, A. R. (1993). Reg – a generalised linear models program, *Technical report*, NSW Department of Agriculture, Sydney.

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 data using a generalized linear mixed model, *Journal of Animal Breeding and Genetics* **104**: 149–155.

Index