ASReml®

What's new in Release 4.3

A R Gilmour

VSN International, Hemel Hempstead, United Kingdom

R Thompson

Rothamsted Research, Harpenden, United Kingdom

March 2026

What's new in Release 4.3

The following significant changes have been made in ASReml 4.3

- Genetic model effects trimming has been implemented for MET analyses, which results in reductions of computing time (Section 4.4)
- ASReml has incorporated some enhancements into factor analytic models that make them run faster (Section 3.3).
- An eigen-model transformation has been implemented that is relevant to incorporating a GRM, where this matrix is transformed to a diagonal structure that is more efficient (Section 4.3).
- New functionalities to construct genomic matrices have been added, including forming an ***H*** (hybrid) matrix and its inverse (Section 4.5), and constructing dominance and epistatic matrices from marker data (Section 4.2).
- Additional qualifiers have been added, particularly associated with generating and saving matrices, such as the full coefficient matrix and its inverse (Section 1.2), and binary forms of the GIV/GRM (Section 4.1)

Contents

1. Changes Associated with Job Control

The following is a list of qualifiers that have been added or modified. These changes are associated with Chapter 5 in the ASReml User Guide (Release 4.3).

1.1 General Job Control Qualifiers

These qualifiers must be placed between the data file line and the model line.

`!DEFINE` *s t p*

Formerly `!CONTRAST`, it provides a convenient way to define how particular effects, 'contrasts' among treatment levels, are defined. `!DEFINE` lines occur as separate lines between the datafile line and the model line. For example

```
!DEFINE LinN Nitrogen 3 1 -1 -3
```

defines `LinN` as a 'contrast' based on the 4 (implied by the length of the list) levels of factor `Nitrogen`. Missing values in the factor become missing values in the 'contrast'. Zero values in the factor (no level assigned) become zeros in the 'contrast'.

`!NOZEROVC`

It suppresses the default action of fixing simple variance components exactly at 0.0 rather than fixing them at a small value. The default action applies to simple components, such as elements in a `diag()` structure and Ψ values in the extended factor analytic models.

`!S2LIMIT`

It limits the size of the residual variance in a multi-environment spatial analysis to a tenth of the average spatial residual variance across all environments, fixing it at that small but not unreasonable value if the updated value is smaller. The motivation is that occasionally in unreplicated trials with a nugget variance also fitted, the spatial variance is poorly estimated and will otherwise go to almost zero, which may cause ASReml to fail.

1.2 Qualifiers Associated with a Coefficient Matrix and Related Matrices

ASReml 4.3 has incorporated a few qualifiers to generate and save the coefficient matrix C and its inverse. All forms are typically sparse, and only the known non-zero cells are written.

`!CINV [`*t*`]`

It writes the known cells in the sparse portion of the C inverse, pertaining to model term *t*, to *basename*`.cii`. The known cells are primarily those that were not zero before inversion. The `.cii` file is ASCII sparse stored with *row column value* for each known cell, lower triangle row-wise.

`!CMAT`,
`!CPMAT`,
`!CAMAT`,
`!CIPMAT`,
`!CIMAT`,
`!DOUBLE`

These output matrices were added to facilitate research into efficient processing of the C matrix of the mixed model equations. They write five forms of C as follows

`!CMAT` writes *basename*`_Cmat.bin`, C in model order,

`!CPMAT` writes *basename*`_CPmat.bin`, C permuted to analysis order,

`!CAMAT` writes *basename*`_CAmat.bin`, C absorbed in analysis order,

`!CIPMAT` writes *basename*`_CIPmat.bin`, the sparse C^{-1} in analysis order,

	`!CIMAT` writes *basename*`_CImat.bin`, the sparse $\boldsymbol{C}^{-1}$ in model order,
	`!DOUBLE` stores the column numbers and matrix values in double precision.
	The file extension is then `.dbin`.

The ASReml User Guide (Release 4.3) discusses the format of these files and how to read them.

1.3 Deprecated and Superseded Functions & Qualifiers

The following qualifiers are now deprecated and superseded.

`!CONTRAST`	This has been renamed `!DEFINE` because the variate inserted in the model is not necessarily a formal treatment (sum to zero) contrast, and the fitted effect is not necessarily the contrast that may be expected.
`!VCC` *n*	Its functionality has been replaced with the `VCC` directive used for linking variance parameters. Details are presented below.

2. Changes Associated with Specifying terms in the Mixed Model

The following is a new additional reserved word. These changes are associated with Chapter 6 in the User Guide (Release 4.3).

`half`	It is used in the linear model to include a column with a coefficient of 0.5. It is primarily used to scale interacting terms as in `half.sire` and `half.dam`.

The following qualifier is now modified.

`!WT`	Used for weighted analyses as a qualifier to the response variable. Its default and specification have changed to rationalize the specification of scaling factors in weights and dispersion factors in GLMM.

3. Changes Associated with Variance Structures

These changes are associated with Chapter 7 in the User Guide (Release 4.3).

3.1 Using !VPGROUP to Constrain Variance Components

There is an additional qualifier `!VPGROUP f1 [f2] [f3]` that facilitates the constraints of variance parameters in `sat()` terms to common values.

For the residual model

```
sat(Exp).ar1(Row).ar1v(Col)
```

when variance parameters associated with experiments can be classified into *groups* the user may wish to constrain the variance parameters to be the same for *experiments* in the same *group*. `!VPGROUP` may have up to 3 arguments, which are applied in order.

For example,

```
sat(Exp !VPGROUP Group1 Group2 Group3).ar1(Row).ar1v(Col)
```

where `Groupi` (i = 1, 2, 3) are variables that group the data associated with `Exp`. This example has 3 spatial parameters for each experiment in the order: row correlation, column correlation and variance; ASReml will then constrain the i-th parameter according to `Groupi`.

For example,

```
sat(Exp !VPGROUP mu Group Exp).ar1(Row).ar1v(Col)
```

allows a common row autocorrelation across all sections, a field specific column autocorrelation and a different variance for each section. If the same grouping is applied to all parameters, the specification can be reduced to

```
sat(Exp !VPGROUP Group1).ar1(Row).ar1v(Col)
```

This qualifier can also be used with `sat()` and `grm()`, among others.

3.2 Simple Relationships Among Variance Structure Parameters: VCC

Formal linear relationships between variance structure parameters can be defined by placing the `VCC` directive after the residual line. Unlike the case of parameter equality, all parameters can be accessed, and the linear relationship is not limited to equality. Since the parameter positions (which are given in the `.tsv` file) are not easily anticipated, the `VCC` statement begins by identifying the first model term containing a parameter to be linked to other parameters.

The syntax is

`VCC` *term* `!LINK` *relative position* `[!SCALE` *scales*`]` `[!BLOCKSIZE` *s*`]`

or

`VCC` *termA* `[-]` *termB*

- *term* is the model term containing the first parameter to be linked,

- *termA* is a model term with a single variance parameter,
- *termB* is another model term with a single variance parameter, which is to be made equal to that for *termA*, or the same but with opposite sign.

In addition, we have

`!LINK` *relative positions*	This is a list of parameter positions where 1 is the first parameter in *term*. For equally spaced positions, use ':' to represent positions between the second and the last; the increment is taken from the initial interval.
`!SCALE` *scales*	Sometimes, the relationship required among linked parameters is not equality. Hence, supply the relative size coefficient for each member of the `LINK` list.
`!BLOCKSIZE` *b*	It is provided for the situation where there are `g` groups of `b` parameters, and we wish to constrain the first parameter of each group numbered (say, $s_1, s_2, \ldots s_g$) and then constrain the second parameter, the third, and so on. Hence, `!BLOCKSIZE` allows to constrain the first set of parameters, and then ASReml will generate the constraints for the other `b-1` sets.

A couple of examples are presented in the table below.

`VCC` statement	action

`VCC idv(units) !LINK 1 2 !SCALE 1 -1`
`VCC idv(units) !LINK 1 -2`
`VCC idv(units) -uni(Check)`
	The parameter following units is to have the same magnitude but opposite sign. These three lines are equivalent to a model line
	`y ~ mu Cov Ch mv !r,`
	`idv(units !INIT 1),`
	`uni(Check !INIT -1)`
	`residual ar1v(Col):id(Row)`

`VCC us(Env).idv(blocks) !LINK 2 4 5 7 8 9`	For a (4 × 4) `US` matrix `us(Env)`, the covariances are made equal.

For other complex constraints that relate to the `sat()` structures, the `!VPGROUP` qualifier can be used (see above).

3.3 Enhancements on Factor Analytic Models

ASReml 4.3 has the structures `fak()`, `facvk()`, `xfak()`, `rrk()`, and `rrdk()`, which are different parameterizations of the factor analytic model in which $\boldsymbol{\Sigma}$ is modelled as $\boldsymbol{\Sigma} = \boldsymbol{\Gamma}\boldsymbol{\Gamma}' + \boldsymbol{\Psi}$ where $\boldsymbol{\Gamma}^{(\omega \times k)}$ is a matrix of loadings on the covariance scale and $\boldsymbol{\Psi}$ is a diagonal vector of specific variances. The structure `rrdk()` has been added to ASReml 4.3.

It is not unusual for users to have trouble comprehending and fitting extended factor analytic models, especially with more than two factors. Two relevant structures are presented below.

`rrk(rr` for *reduced rank*)	This is formally just an alternate specification of `xfak()` but with the specific variances ($\boldsymbol{\Psi}$) all set to zero, which means that if you supply initial values, you need to supply initial values for the specific variances as zero.

`rrdk (rr` for *reduced rank* and `d` for *diagonal*)	This is equivalent to `xfak()`, but fits it as two independent model terms: an `rrk()` term and a `diag()` term. This form of the mixed model equations runs faster with large relationship matrices. For example
	`rrk(trial).grm(entry) + diag(trial).grm(entry)`
	will run faster than
	`xfak(trial).grm(entry)`

The structure `rrdk()` is new to ASReml 4.3. One advantage of the `rrdk()` structure is that it runs faster than `xfak()` when interacting with a GRM matrix. We also denote `rrdk()` as a RRD (reduced rank + diagonal) model.

3.4 Deprecated and Superseded Qualifiers

The following qualifiers are now deprecated and superseded.

`!OFFSET` *o*	It is a deprecated qualifier associated with `!VCC`. It is superseded by new arguments in `VCC`.
`!SUBSECTION` *f*	For this `RESIDUAL` line qualifier, *f* is a factor in the data that breaks the section into independent subsections, with subsections having common variance parameters. It is now deprecated and replaced by `sat(!VGROUP)`.

4. Changes Associated with Genetic Analyses

These changes are associated with Chapter 9 in the User Guide (Release 4.3). Some new qualifiers for the GRM/GIV and GRR lines have been added to enable the saving and keeping of relationship matrices and the formation of other relationship matrices. This section summarizes the qualifiers, and later sections discuss the new relationship matrices in more detail.

4.1 Summary of New Qualifiers Associated with Relationship Matrices

`!DOMINANCE`	This qualifier used on the GRR line invokes the formation of a dominance matrix from marker data ($\boldsymbol{G_D}$). Further details are in Section 4.2.
`!EIGTRANSFORM`	This qualifier used on the GRM line invokes a singular value decomposition (SVD) of $\boldsymbol{G}$ ($\boldsymbol{G = UDU'}$ with $\boldsymbol{UU' = I}$ and $\boldsymbol{D}$ diagonal), holds the $\boldsymbol{U'}$ and $\boldsymbol{D}$ for subsequent use, and writes them to files. Further details are in Section 4.3.
`!EPISTATIC`	This qualifier used on the GRR line invokes the formation of epistatic matrices from marker data. Further details are in Section 4.2.
`!HINV`	This qualifier used on the GRM line invokes the inverse of a $\boldsymbol{H}$ (or hybrid) matrix by merging the inverse of an $\boldsymbol{A}$ (numerator relationship) matrix with the inverse of a $\boldsymbol{G}$ (genomic relationship) matrix pertaining to a subset of the genotypes in $\boldsymbol{A}$ (Legarra *et al.*, 2009). Further details are in Section 4.4 of this and associated qualifiers.
`!KEEPGRM`	This GIV/GRM qualifier instructs ASReml to keep in memory the GRM matrix as well as its inverse. Further details are in Section 4.6, together with its use in conjunction with `sat()` and its qualifier `!VGROUPS` in the formation of trimmed genetic relationship matrices.
`!SAVEGIV [`f`]`	This pedigree qualifier is used to write the $\boldsymbol{G^{-1}}$ matrix in binary form so that it can be read back in ASReml, and so save re-computation and have reduced reading time. This facility has been extended to pedigree and GRR lines. If f = 3, the inverse matrix is written as a binary `.sgiv` file in single precision, and f = 4 writes the inverse matrix as a binary `.dgiv` file in double precision. If used with `!KEEPGRM` on the GIV/GRM lines, the uninverted $\boldsymbol{G}$ matrix is also written to `.sgrm`/`.dgrm` binary files. Further details are in Section 4.6 of binary matrices in ASReml.

4.2 Dominance and Epistatic GRM Matrices for Marker Data

GRM matrices are typically formed from marker matrices ($\boldsymbol{M}$) in which SNPs are coded 0, 1 or 2 and arranged with genotypes in rows (lines of the file) and SNPs as data fields. Vitezica *et al.* (2017) define

$$\boldsymbol{G_A = MM'/(2\sum_{j=1}^{m} p_j(1-p_j)}$$

where $\boldsymbol{M}$ has elements (0 - $2p_j$), (1 - $2p_j$) and (2 - $2p_j$) for genotypes AA, AB and BB, respectively, $2p_j$ being the mean for SNP j.

$$\boldsymbol{G_D = KK'/(4\sum_{j=1}^{m} p_j^2(1-p_j^3)}$$

where $\boldsymbol{K}$ has elements $-2p_j^2$, $2p_j(1 - 2p_j)$ and $-2(1-p_j^2)$ for genotypes AA, AB and BB, respectively. Also,

$$\boldsymbol{G_{AA} = G_A \# G_A/(tr(G_A \# G_A)/n)}$$
$$\boldsymbol{G_{AD} = G_A \# G_D/(tr(G_A \# G_A)/n)}$$
$$\boldsymbol{G_{DD} = G_D \# G_D/(tr(G_D \# G_D)/n)}$$

where # represents the Hadamard product and tr() is the trace function.

The matrices are available in ASReml with the use of the `!DOMINANCE` and `!EPISTATIC` qualifiers on the `.grr` file specification line, which supplies the file containing the marker (0, 1, 2) matrix, where `!DOMINANCE` requests $\boldsymbol{G_D}$ be formed (along with $\boldsymbol{G_A}$), and `!EPISTATIC` requests $\boldsymbol{G_{AA}}$ be formed (also $\boldsymbol{G_{AD}}$ and $\boldsymbol{G_{DD}}$ if $\boldsymbol{G_D}$ is available).

4.3 Eigen-Model Transformation with `!EIGTRANSFORM`

In the context of needing to fit a large genomic *Animal* model to many traits, an eigenvector transformation of the model design matrix can result in faster processing (for more details, see Lee and van der Werf 2016). This idea is relevant when the GRM term dominates the model. It essentially turns the part of the coefficient matrix relating to genomic effects to a diagonal structure at the expense of turning the (much fewer) non-genomic effects into dense equations.

The `!EIGTRANSFORM` qualifier on the GRM line invokes a singular value decomposition (SVD) of $\boldsymbol{G}$ ($\boldsymbol{G = UDU'}$ with $\boldsymbol{UU' = I}$ and $\boldsymbol{D}$ diagonal), holds the $\boldsymbol{U'}$ and $\boldsymbol{D}$ for subsequent use and writes them to file (..._`D.sgrm`, ..., _`U.sgrm`). If the $\boldsymbol{U'}$ and $\boldsymbol{D}$ files already exist, $\boldsymbol{U'}$ and $\boldsymbol{D}$ are retrieved from a file. Then, if this GRM is specified in the model, and the model and data meet the necessary requirements, the model design matrix is transformed (pre-multiplied by $\boldsymbol{U'}$) except for the columns specific to the GRM; $\boldsymbol{D}$ is used as the GRM variance matrix for the genetic effects in the analysis of the transformed model so that this block of the $\boldsymbol{C}$ matrix remains diagonal. Generally, this will run much faster.

The eigen-transformation is only possible when there is one data record for each genotype, as occurs with the *Animal* model. A large amount of workspace is required for the process of transforming the design matrix.

The fitted effects between the two models agree except for the genomic BLUPs. The BLUPs from the conventional BLUP ($\boldsymbol{u}$) are calculated as $\boldsymbol{u = Us}$, where $\boldsymbol{s}$ are the BLUPs from the transformed analysis. ASReml 4.3 does not calculate them at present.

For example, for the common genomic animal model

```
!WORK 6
 10K bivariate data
 ID !A !LL20 !L ID_order.txt
 CG * CGs *              # contemporary group factors
 imf sf5                 # response variates
 A.sgiv !GDENSE          # A inverse genomic matrix
 data.csv !SKIP 1        # data file in same order as A22
 imf ~ CG !r grm1(ID)
```

the equivalent model and two other similar equivalent models can be fitted with

```
!WORK 6 !RENAME 1 !ARG 1 2 3 !DOPART $1
10K bivariate data
 ID !A !LL20 !L IDorder.txt
 CG * CGs *              # contemporary group factors
 imf sf5                 # response variates
 tCG !G 376              # CG design matrix
 A.grm !EIGENTRANSFORM   # instruction for eigen value analysis
 data.csv !SKIP 1        # data file

!PART 1
imf ~ CG !r grm1(ID)     # transformed model
!PART 2
sf5 ~ CG !r grm1(ID)     # transformed model
!PART 3
imf sf5 ~ Trait !r us(Trait).grm1(ID) !f Trait.CG # bivariate transf. model
```

The `!EIGTRANSFORM` qualifier performs an SVD (Singular Value Decomposition, eigen analysis) of the ***G*** matrix and holds the eigenvalues and vectors. It also flags that when the model is fitted, the design matrix is to be transformed using the eigenvectors.

This approach only works when the data file has the same (or fewer) rows as the GRM matrix and the number of other effects in the model is substantially less than the number of genotypes. It will save considerable time when there are many response variates with no missing values to analyse, especially for bivariate analyses.

Comparing the two models given in the code above, with 9,688 genotypes, the conventional analysis took 9 sec (to invert GRM) + 8×43 sec (for 8 iterations). The overhead of the SVD factorization was 160 sec and 8 sec (for transforming the model); the transformed analysis took 8×2 sec for 8 iterations. A conventional bivariate analysis took 8×313 sec. The transformed bivariate analysis took 8×13 sec, a considerable difference.

The fitted effects between the two models agree except for the genomic BLUPs. As indicated before, the BLUPs from the conventional BLUP (***u***) are calculated as $\boldsymbol{u} = \boldsymbol{Us}$, where ***s*** are the BLUPs from the transformed analysis.

4.4 Genetic Trimming in MET Models with `!KEEPGRM`, `sat()` and `!VGROUP`

The situation where not all genotypes are observed in all environments is becoming more common. That is, when each environment (`Env`) samples different subsets of the genotypes (`Gen`). This, when combined with the use of a GRM matrix to specify the genetic relationships, can quickly result in a large and fairly dense set of equations to solve.

The approach presented here is motivated by the work of Mazur (2021) and concerns the fitting of genotype by environment factor analytic models with a dense `grm(Gen)` matrix term. As mentioned in the previous section, `xfa`*`k`*`(Env).grm(Gen)` can be more efficiently fitted as `rr`*`k`*`(Env).grm(Gen) + diag(Env).grm(Gen)`. Mazur (2021) suggested replacing `diag(Env).grm(Gen)` by a set of separate model terms, one for each level of `Env`, using only the subset of genotypes with data for that level (*i*) of `Env` and using the associated reduced GRM matrices. The author calls this operation *trimming*, and this leads to the same variance model (hence, identical variance parameter estimates in the two models).

The expanded model limits the prediction of genotypes to the environments where they occur, and this can result in a dramatic reduction of computing time, as a result of creating a separate model term for each environment using a reduced environment-specific GRM matrix. In particular, you may just want an average genotype effect (across environments) or relative genotype effects for only those genotypes present at a particular environment.

This operation of forming separate variance structures for each level of environment is analogous to the operation at the residual level of forming similar variance models for each level of `sat()`, but the sizes are defined by the data associated with each level of `sat()`. ASReml 4.3 uses

```
sat(Env).grm(Gen)
```

to specify this operation. Each of the expanded terms is denoted in the output file as

```
sat(Env,i).grm(Gen|Env_i)
```

where `i` indexes the environment, `grm(Gen|Env_i)` denotes the reduced GRM matrix specific for the genotypes present at environment `i`. With this model, the `rrk(Env).grm(Gen)` term fits the common (factor) effects and the `sat(Env).grm(Gen)` terms are the environment-specific 'specific' variances (genetic deviation from the common effect).

In order to form the reduced GRM matrices, the full GRM matrix needs to be available, and the GRM qualifier `!KEEPGRM` is used for this.

In some analyses, experiments are grouped, and the same subset of genotypes is used in all experiments in a group. This means that a smaller set of reduced GRM (and GIV) needs to be formed. The qualifier `!VGROUP` *`Group`* has been introduced to allow that.

For example

```
sat(Exp !VGROUP Group).grm(Gen)
```

The nested association between `Exp` and `Group` is deduced from the data, and the generated model terms are denoted in the output file as

```
sat(Exp,i).grm(Gen|Group_j)
```

where `Gen|Group_j` is the '*subset*' of genotypes associated with group `j`, and is fitted using a reduced GRM pertaining to those genotypes in the data pertaining to experiment `i`, which is a member of group `j`.

We can also use `!VPGROUP` to constrain the variance parameters for a parameter-reduced version of this model. Consider the untrimmed model

```
Yield ~ mu Exp !r Rep.Exp Block.Rep.Exp grm(Gen) idv(Exp).grm(Gen)
```

This model uses the same variance parameter for all `Exp.Geno` effects. A trimmed version of this model is

```
Yield ~ mu Exp !r Rep.Exp Block.Rep.Exp grm(Gen) sat(Exp !VPGROUP mu).grm(Gen)
```

with `!VPGROUP mu` constraining the parameters associated with `Exp.Gen` to be the same.

There is a more detailed wheatgrass example in Section 16.8.3 of the User Guide (Release 4.3). In this example, the trimmed model took 33.3 seconds per iteration compared with 116.7 seconds per iteration for the original model. More detailed timings are in Section 3.2 below, which are consistent with the findings of Mazur (2021).

4.5 Forming the *H* matrix

The $\boldsymbol{H}$ (or hybrid) matrix is a particular form of a $\boldsymbol{G}$ matrix obtained by merging an $\boldsymbol{A}$ (numerator relationship) matrix with a $\boldsymbol{G}$ (genomic relationship) matrix pertaining to a subset of the genotypes in $\boldsymbol{A}$ (Legarra *et al*., 2009).

$$\boldsymbol{H}_{\tau,\omega}^{-1} = \boldsymbol{A}^{-1} + \begin{pmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & (\tau\boldsymbol{G}^{-1} - \omega\boldsymbol{A}_{22}^{-1}) \end{pmatrix}$$

where $\boldsymbol{A}_{22}^{-1}$ is the inverse of the portion of the $\boldsymbol{A}$ matrix that contains the genotyped individuals. Note that the cells present in the $\boldsymbol{H}$ inverse can be tuned by ω and **τ,** which have default values of 1 (for more details, see Martini *et al.* 2018).

The qualifiers

```
!HINV GRM_ID_file.txt [!HSKIP h] [!OMEGA ω] [!TAU τ]
```

on a GRM definition line forms the special $\boldsymbol{G}$ inverse known as an $\boldsymbol{H}$ inverse. A pedigree from which to form an $\boldsymbol{A}$ inverse must have already been specified. The genotype identifiers in the $\boldsymbol{G}$ matrix are to be specified as a list in the ASCII file provided as the argument to the `!HINV` qualifier. All identifiers must also be present in the pedigree.

In order to form the $\boldsymbol{H}^{-1}$ matrix, the pedigree file from which the $\boldsymbol{A}^{-1}$ matrix is first formed, then the $\boldsymbol{G}$ (or its inverse) matrix is specified and read, and finally, the `!HINV` qualifier forms the $\boldsymbol{H}^{-1}$ matrix. Note that the GRM line with the `!HINV` qualifier, defines two inverse matrices available for use in the model: the $\boldsymbol{G}^{-1}$ and the $\boldsymbol{H}^{-1}$. Therefore, we could fit any of `nrm(ID)`, `grm1(GID)` or `grm2(ID)` in the model. This assumes that the levels associated with the `!HINV` qualifier are included in the pedigree.

ASReml saves the $\boldsymbol{H}^{-1}$ matrix as a binary file (filename given in the output) which can subsequently be used directly (saving the setup time in the subsequent runs).

4.6 Dealing with Binary *G* Matrices

ASReml 4.3 can read in a GRM matrix in various forms defined by content ($\boldsymbol{G}$ or $\boldsymbol{G}^{-1}$), form (ASCII, REAL_S or REAL_R) and layout (row-wise or cell-wise). Their content and form are indicated by the following filename extensions

content	ASCII	REAL_S	REAL_D	REAL_R
$\boldsymbol{G}$	`.grm`	`.sgrm`	`.dgrm`	`.rgrm`
		`.bgrm`		
$\boldsymbol{G}^{-1}$	`.giv`	`.sgiv`	`.dgiv`	`.rgiv`
		`.bgiv`		

The binary forms are preferred because the files are smaller and accuracy is greater. Providing the inverse and log-determinant of the matrix saves processing time calculating them. REAL_S refers to the FORTRAN sequential binary file structure in which each 'record' is enclosed in a 'wrapper' indicating the record size in bytes. When ASReml writes a binary file, it uses the REAL_S file

structure. REAL_D is a double-precision binary file, but the extra precision is generally unnecessary. REAL_R refers to the R(C) binary form, which does not include any 'record size' information.

Typically, $\boldsymbol{G}$ and $\boldsymbol{G}^{-1}$ are dense matrices ((nearly) all cells non-zero) and are half-stored. However, some very large matrices have significant sparsity (most cells are zero).

Note that

- `.sgiv`, `.sgrm`, `.bgrm`, `and .bgiv` files are binary, single precision 'sequential' files containing sparse or dense matrices of various styles.
- `.rgiv` and `.rgrm` files are 'C-style' binary, single precision, written by R as binary versions for the ASCII layouts.
- `.dgiv` and `.dgrm` files are binary, dense format double precision files.

5. Changes Associated with PREDICT and TABULATE directives

The following are qualifiers for the PREDICT directive. These changes are associated with Chapter 10 in the User Guide (Release 4.3).

!ESTIMATE [*coef*]

This PREDICT qualifier instructs ASReml to form the full variance/covariance matrix for the predicted values and then calculate a series of linear functions across the set of predictions. For example, if we have a factor SNP with 3 classes AA, AB, BB in that order,

```
PREDICT SNP !ESTIMATE -1 0 1 -1 2 -1
```

would report 2 contrasts over the 3 predicted values.

!PAC [*t*]

(Predict Actual Coefficients) is used instead of !ONLYUSE *t* when *t* is a design function of a variable (like leg(), mbf() or spl()), and the prediction pertains to the actual coefficients, not the function of the coefficients, as in

```
!MBF mbf(times,12) MyZeds.txt !SKIP 1 !KEY 1 !RENAME Zeds
accel ~ mu times !r Zeds
PREDICT Zeds 1 2 3 4 5 6 7 8 9 10 11 12 !PAC Zeds !VPV
```

where times is a covariate and MyZeds.txt has a leading key variable listing the unique times followed by 12 base variables. With !PAC also specified, the (default) predict points 1:12 (in the PREDICT line) index the base variables rather than being covariate values for times. Job output qualifier !CINV Zeds is another way of getting the variance matrix for the 12 effects.

!SPLIT

This PREDICT qualifier causes the predict output from each prediction to be split into multiple files for easier parsing. The default .pvs file holds all explanatory text with the prediction tables. With !SPLIT, the predicted value lines are reported in an *basename*_PVT_ii.txt (.csv) file in a form easily loaded into Excel or R. All explanatory text associated with the predict table are written to *basename*_PVH_ii.txt file where ii is the index of the PREDICT statement.

!TDIFF

This PREDICT qualifier requests *t*-statistics be printed for all combinations of predicted values. It creates a pairwise table of *t*-statistics among all predicted values (assuming the number of cells predicted is not ridiculously large). ASReml reports Bonferroni and Tukey critical values for assessing the significance of the *t*-statistics in some common situations. For a 1-way table, the denominator degrees of freedom for the factor need to have been calculated for the Wald F-table. For a 2-way table, the variance model needs to be simple, such as a split-plot, in which stratum variances as well as denominator degrees of freedom are available.

The following is a qualifier for the TABULATE directive.

!PRINTALL

This qualifier reports empty cells in the tabulation. By default, such cells are not reported. This is useful for identifying where empty cells appear in the class list.

6. Software Performance

Timing information useful for comparing execution time between models and/or builds of ASReml is available in the `.asl` file if the `!LOGFILE` and `!DEBUG` qualifiers (or command line options `-DL`) are set on the first line of the job (above the TITLE line). Running the command `grep '>>' job.asl` at the command prompt will extract timing information. (`grep` is a UNIX/Linux command used to find a specific string from inside a file. For Windows, the `grep` alternative is `findstr`.

We give two examples of timings using ASReml 4.3. The first example is of the comp of the comparison of `XFA` and `RRD` models as discussed in Section 3.3. The second example is of trimming discussed in Section 4.4.

6.1 XFA/RRD Example

This is a complex multi environment analysis with 24 experiments and 884 genetic lines linked with a ***G*** matrix for a total of 12,354 records. The model fitted is

```
yield ~ mu Experiment,
!r Rep.Experiment Block.Rep.Experiment grm(Line).xfa1(Experiment)
```

and it contains 22,797 equations. The log-file output for this analysis is

```
 >> Windows x64   30.0 Gbyte  PhenoG2F8.ni/PhenoG2F  17 Apr 2025 08:59:00.497
 >> >>      ASReml Process        Clock   SumClock
 >> >>     Getting Started: sec       1.04       1.04
 >> >>       Before Order : sec       0.20       1.24
 >> >>        * Ordering : sec       2.23       3.48
 FILLIN 26879705 ==>> 27580197    1.03, #SR: 22772, AvLen: 1211, MxLen: 2023, AvFlops: 818711
 >> >>         C reordered: sec       4.63       5.87
  >> Cabs blocks          407   49.93858             95
 >> >>          C absorbed: sec       3.32       9.19
 >> >>           WV formed: sec       0.06       9.25
 >> >>           AI formed: sec       0.38       9.63
 >> >>           Ci formed: sec       4.22      13.86
         373  >> Ci nodal blocks; average    55.13137
 >> >>        Ci reordered: sec       1.68      15.54
 >> >>         Gscore done: sec       1.20      16.75
 >> >>  Iteration complete: sec       0.02      16.77
 >> >>  Iteration complete: sec      13.24      30.01
 FILLIN 25780785 ==>> 26263304    1.02, #SR: 22772, AvLen: 1153, MxLen: 1781, AvFlops: 772350
 >> >>     Iterations done: sec     152.06     182.07
 >> >>         SLN written: sec       0.12     182.19
 >> >>         YHT written: sec       0.07     182.26
 >> >>            Finished: sec       0.00     182.26
```

This job required 13 iterations in 182 seconds (that is, 13.2 seconds per iteration), and had a log-likelihood value of -46,808.1.

In ASReml 4.3, it is now possible to fit the following equivalent model

```
yield ~ mu Experiment,
!r Rep.Experiment Block.Rep.Experiment,
rr1(Experiment).grm(Line) diag(Experiment).grm(Line)
```

that, in contrast, required 3.3 seconds per iteration, for a total of 15 iterations with a very similar log-likelihood value of -46,810.2.

```
>> Windows x64   30.0 Gbyte  PhenoG2F9/PhenoG2F  17 Apr 2025 10:28:11.087
>> >>      ASReml Process        Clock   SumClock
>> >>     Getting Started: sec       1.03       1.03
>> >>       Before Order : sec       0.07       1.10
>> >>        * Ordering : sec       1.22       2.32
FILLIN 9391895 ==>> 10424393    1.11, #SR: 43988, AvLen: 237, MxLen: 862
>> >>         C reordered: sec       1.69       2.79
 >> Cabs blocks          292   73.28082             95
>> >>          C absorbed: sec       0.70       3.49
>> >>           WV formed: sec       0.20       3.68
>> >>           AI formed: sec       0.12       3.81
>> >>           Ci formed: sec       0.70       4.51
        233  >> Ci nodal blocks; average    88.08154
>> >>        Ci reordered: sec       0.38       4.89
>> >>         Gscore done: sec       0.77       5.66
>> >>  Iteration complete: sec       0.01       5.67
>> >>  Iteration complete: sec       3.29       8.96
>> >>     Iterations done: sec      44.33      53.29
>> >>         SLN written: sec       0.22      53.51
>> >>            Finished: sec       0.07      53.58
```

Typically, as shown above, the major components are: `Order found`, `C absorbed` and `Ci formed`. The `FILLIN` line reports the size of the ***C*** matrix before and after absorption, and the ratio (after/before) of the sizes.

6.2 Genetic Model Trimming Example

Consider the analysis of a breeding experiment for wheatgrass, a perennial wheat species. The analysis is of 22 harvests taken from 10 trials planted over 4 years. The selection of genotypes planted in a trial differs considerably, the later plantings being lines (genotypes) derived from earlier plantings. A simplified model for the trait plant height, `PTHT`, using a genotype relationship matrix (marker-based) to provide genetic links is

```
PTHT ~ mu mv !r Harvest rr1(Harvest).grm1(geno) at(Harvest).grm1(geno) female.male
residual sat(Harvest).ar1v(row).ar1(range)
```

This corresponds to a factor analytic model where the term provides the overall genetic covariances while the `at(Harvest).grm1(geno)` term provides the specific genotype by harvest variance components for each harvest.

The coefficient matrix includes 23 instances of the full genomic matrix, which is of order 4,482. An alternative way of setting up the model is just to include the genotypes with data in the GRM used for estimating the 22 specific variance components. The matrices vary in size from 308 to 576 genotypes. The model in this case is

```
PTHT ~ mu mv !r Harvest rr1(Harvest).grm1(geno) sat(Harvest).grm1(geno) female.male
residual sat(Harvest).ar1v(row).ar1(range)
```

The variance components from this model are the same as for the full model, but this model took 33.3 seconds per iteration compared with 116.7 seconds per iteration for the original model.

Comparison of the model processing details is presented in the table below.

action	using `at()`	using `sat()`
Initialize (sec)	1	1
Ordering (sec)	50	7
Absorption (sec)	43	4
Forming AI (sec)	13	13
Sparse Inverse (sec)	40	4
Score (sec)	16	13
Equations	367,587	278,397
C matrix cells	214,560,191	12,453,485

7. References

Lee, S. H. and van der Werf, J. H. J. (2016). MTG2: an efficient algorithm for multivariate linear mixed model analysis based on genomic information. *Bioinformatics* **32**(9): 1420-1422.

Legarra, A., Aguilar, I, and Misztal, I. (2009). A relationship matrix including full pedigree and genomic information. *Journal of Dairy Science* **92**(9): 4656–4663.

Martini, J. W., Schrauf, M. F., Garcia-Baccino, C. A., Pimentel, E. C., Munilla, S., Rogberg-Muñoz, A., Cantet, R. J., Reimer, C., Gao, N., Wimmer, V., and Simianer, H. (2018). The effect of the H^{-1} scaling factors τ and ω on the structure of H in the single-step procedure. *Genetics Selection Evolution* **50**(1): 1–9.

Mazur, L. (2021). Computational methods for the fitting of factor analytic linear mixed models with applications to plant variety trials. PhD thesis, University of Wollongong, Australia.

Vitezica, Z. G., Legarra, A., Toro, M. A., and Varona, L. (2017). Orthogonal estimates of variances for additive, dominance, and epistatic effects in populations, *Genetics* **206**(3): 1297-1307.