4.9.6 Relationships between variance parameters

Equality relationships among paremters in a single variance structure may be specified using the `!=` qualifier (Section 4.9.5). More general constraints may be defined using *constraint lines*.

- The number of constraint lines is specified by the `!VCC` qualifier described in Section 4.7.2.
- Constraint lines must follow the definition of all **R** and **G** structures.
- Each constraint is specified on a separate line in the form
  $+ \ P_1 * V_1 \quad P_2 * V_2 ...$
  where P_i is the number of a parameter and V_i is a coefficient.
  If the coefficient is 1., you may omit '* 1';
  if the coefficient is -1, you may write $-P_i$ instead of $P_i * -1$. The meaning of the coefficients is a follows: $P_2 = V_2 * P_1/V_1$. Typically V_1 is 1.
- For example
  `+ 5 7 * .1` means parameter 7 is a tenth of parameter 5;
  `+ 5 -7` means parameter 7 is the negative of parameter 5.
- The P_i terms refer to positions in the full variance parameter vector. This will change if the model is changed. The full vector includes a term for each factor in the model and then a term for each parameter defined in the **R** and **G** structures. A list of P_i numbers and their initial values is returned in the `.asl` file to help you check the numbers.

4.10 Prediction

This section describes `PREDICT` directives which may be appended to the job. This is a new facility under development since August 2000. It provides an easier mechanism for obtaining predicted values (least squares means) than the `.pin` file mechanism described in chapter 6. However, development of a robust procedure to handle the range of models available in **ASReml** is not trivial. At this stage only simple cases are covered. Further development is planned but this release is to enable user's to give feed back on problems and unmet expectations.

Important
Disclaimer: Please double check the results obtained using this predict facility. **ASReml** handles an extensive range of mixed models and your particular model may be one which is not adequately handled by the implementation so far. Please send any jobs that appear to fail to arthur.gilmour@agric.nsw.gov.au as part of the Beta testing process.

The `PREDICT` statement must appear as the final section of an **ASReml** job after any **R** and **G** structure lines. It consists of

- the directive `PREDICT` starting in column 1 of the line.
- a list of the factors defining a multiway table to be predicted. Each factor may be followed by a list of specific values at which the prediction is to occur.
- a list of qualifiers.

Several predict statements may be specified.

ASReml parses the `PREDICT` directive before fitting the model. Any syntax problems encountered are reported in the `.asr` file but do not cause the job to fail. **ASReml** undertakes the prediction as part of the final, but not the first, iteration. I.e. **ASReml** must do at least two iterations. On the first it collects information about the model which it needs when setting up the prediction design matrix. It then reports the values to the `.pvs` file. Aborting a run by creating the `ABORTASR.NOW` file will cause any predict statements to be ignored.

Since prediction is often desired after exploring a range of models, one approach is to do the modelling without worrying about `PREDICT` statements and then, at the final stage, append the `PREDICT` statements and rerun the job using the c (`!CONTINUE`) command line option.

4.10.1 Limitations

The prediction module cannot (yet) handle the following situations:

- Constrained (`con()`) model terms with singularities. These will always be disallowed because the disentangling of effects is potentially to complex.

- Multiple traits
- Pedigreed factors

4.10.2 Predict factors

The table to be predicted is defined by listing the factors defining its margins. Other factors in the model will be averaged over.

Prediction at particular values is achieved by listing the values after the factor name. Where there is a sequence of values, use the notation a b ... n to represent the sequence of values from a to n with step size $b - a$. The default stepsize is 1 (in which case b may be omitted). : may replace .... An increasing sequence is assumed.

The `PREDICT` statement may be continued on subsequent lines by terminating the current line with a comma.

Examples:

```
yield ~ mu variety !r rep
0
predict variety
```

predicts variety means.

```
yield ~ mu lin(row) variety !r rep
0
predict variety
```

predicts variety means at the average of row.

```
yield ~ mu lin(row) variety  c(rep)
0
predict variety row 2
```

predicts variety means at row 2.

4.10.3 Predict qualifiers

`!IGNORE` *t* may occur once to cause the prediction to ignore model term *t*. This might be used to omit the spline Lack of fit term (`!IGNORE row`) from predictions based on the model `Y ~ mu lin(row) variety !r spl(row) row` as in `PREDICT row !IGNORE row`.

`!FULL` Predictions which are probably incorrect because of aliasing of terms in the model arising from missing cells in interaction tables are normally suppressed. The `!FULL` qualifier causes these values to be printed. They are distinquished from the estimable predictions by the character * replacing a P which indicates the prediction is based on cells which are present.

`!MARGIN` is used when predicting components of a two-way table when some of the cells have no data. **ASReml**takes the cells which are properly predicted and does a marginal main effects analysis to predict the missing cells from marginal effects. It also produces the marginal means. These estimated means are identified by the letter E. At present this returns weighted marginal means and predictions based on such. NB This is restricted to two-way tables.

The following backtransformation qualifiers are automatically set when a GLM link function is specified for the analysis.

`!LOGIT` to backtransform when data analysed on the logit scale.

`!PROBIT` to backtransform when data analysed on the probit scale.

`!INVERSE` to backtransform when data analysed on the reciprocal scale.

`!SQRT` to backtransform when data analysed on the square root scale.

`!COMPLOGLOG` to backtransform when data analysed on the complementary log log scale.

`!LOGN` to backtransform when data analysed on the natural logarithm scale.

`!POWER` p to backtransform when data analysed on the Y^p scale.